Scalable Neural Network Verification against Geometric Perturbations via Hölder Optimisation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural Network (NN) verification methods provide local robustness guarantees for a NN in the dense perturbation space of an input point. A key challenge in this area lies in the scalability of the resulting problem, leading to large models of interest in applications not being addressable by the methods. In this paper we introduce HOVER, a method for the verification of NNs against geometric perturbations, that uniquely employs a Hilbert space-filling construction to reduce multidimensional problems to single-dimensional ones. The underlying Hölder optimisation, which also iteratively refines the estimation for the Hölder constant for constructing the lower bound, theoretically it may converge to a local minimum, thereby resulting in a robustness result being incorrect. However, we show experimentally that this risk can be contained in practice by appropriately devised heuristics in the global optimisation setup. Indeed, unlike recently reported errors from theoretically sound implementations, we found no incorrect result by running the technique on a large set of benchmarks from SoundnessBench and VNN-COMP. To validate the scalability of the approach, we report on extensive experiments on large NNs ranging from Resnet34 to Resnet152 and ViTs. These demonstrate the SoA performance of the approach in evaluating with high reliability the local robustness of large NNs against geometric perturbations on the ImageNet dataset. Beyond image tasks, we show that the method's scalability enables for the first time robustness assessments for large-scale 3D-NNs in video classification tasks against geometric perturbations for long-sequence input frames on Kinetics/UCF101 datasets.

1 Introduction

As well known, Neural Networks (NNs) are inherently vulnerable to adversarial perturbations [1], *i.e.*, their output is often susceptible to fragilities, or attacks, in the neighbourhood of correctly processed inputs. In the context of machine vision models, input perturbations generating such fragilities can take various forms including noise, geometric changes, illumination variations, and beyond. Evaluating the robustness of a model, *i.e.*, the resistance to such vulnerabilities, is particularly important in safety-critical applications.

The area of robustness verification [2] consists of methods providing formal guarantees that a model is locally robust in regions of the input space defined by a test point and a particular perturbation. A well-known difficulty of these methods is their scalability: the problem is theoretically NP-hard [3] and present SoA methods do not scale to neither large models used in applications, nor large inputs, nor large perturbations [4], thereby hindering the application of these methods in applications.

With the exceptions discussed in Related Work (Section 5), methods for the verification of local robustness are sound, *i.e.*, if the method reports that the model is locally robust in a region, that is theoretically guaranteed to be the case. While this result provides theoretical comfort, arguably it

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

is less significant in practice. Firstly, the actual correctness of a specific verification query may be hindered by floating point precision errors at system level as well as other issues [5], as recently evidenced in [6], thereby rendering such guarantees less significant in practice. Secondly, overall considerations on the robustness of a model are derived by analysing thousands or more input points and perturbations of various sizes. It is the aggregation of these results, not a single query, that enables the analysis of a model's robustness as well as the comparison between models.

Therefore, as long as errors can be well-contained in practice, the theoretical soundness of a method appears less essential compared to whether or not the method can scale to models in use in applications. Indeed, adversarial testing is routinely used in applications to evaluate the robustness of large models [7, 8]. Yet, adversarial testing is well-known to fail to identify fragilities in a very large number of cases, potentially instilling a false sense of robustness in the developer.

In this paper, we exploit the observations above to introduce HOVER, a method based on Holder optimisation that exploits dimensionality reduction to scale to large models with hundreds of millions of parameters. HOVER is inspired by recent advanced developments in the area of global optimisation [9–11]. As such, in theory, in line with many global optimisation methods [12] for NN verification, its convergence to the global minimum can be assured only if some appropriate optimisation parameters are chosen. We return to this aspect in Section 5, where we observe that most existing global optimisation-based approaches also have this property.

In practice, we demonstrate that the optimisation problem can be appropriately formulated, resulting in no incorrect results on the large-scale benchmarks that we study, including SoundnessBench, and outperforming in practice all current SoA and theoretically sound methods.

In summary, the paper contributions are as follows:

- We propose HOVER, a global optimisation method for the verification of NNs based on space-filling dimensionality reduction and Hölder optimisation. We provide theoretical conditions for convergence, hence soundness. We illustrate that such conditions are difficult to ascertain in practice but that, experimentally, the potential of error in a single query is well contained. Indeed, no errrors were produced following extensive evaluation including SoundnessBench [6].
- We use HOVER to verify the local robustness of models up to 300M tuneable parameters, including ResNet152 and Vision Transformers for image classification tasks, against geometric properties (rotation, scaling, and translation) on the large-scale ImageNet dataset.
- We use HOVER to verify for the first time the geometric robustness of 3D ResNet models in video classification tasks for streams of $32 \times 3 \times 256 \times 256$ inputs.

We believe these results enable a robustness analysis of large models used in applications that could not be undertaken before other than via adversarial search, which has a much lower confidence rate.

The rest of the paper is organised as follows. In Section 2 we present key notions of use throughout. We present HOVER in Section 3 where we give the technical details of the algorithm. In Section 4, we evaluate HOVER on large NNs for image classification and video classification; we also evaluate the correctness of the implementation empirically on SoundnessBench and additional benchmarks from VNN-COMP [4]. Section 5 discusses related work. We conclude in Section 6.

2 Preliminaries

This section outlines the background concepts and notation that facilitate the exposition of the verification method presented in the next section.

Hölder/Lipschitz constant. A function $f : \mathbb{R}^N \to \mathbb{R}$ is said to be Hölder continuous with exponent $\alpha \in (0, 1]$ if there exists a smallest constant $H \geq 0$, called the Hölder constant, such that for all $x, x' \in [a, b]$, the following inequality holds: $|f(x) - f(x')| \leq H|x - x'|^\alpha$. Lipschitz continuity [13] is a special case of Hölder continuity when $\alpha = 1$, in which case H becomes the Lipschitz constant L. These constants represent the highest rate at which the function can change in the interval.

Hilbert space-filling curve. A space-filling curve [14] is a function $h : \mathbb{R} \to \mathbb{R}^N$ that maps the unit interval $x \in [0, 1]$ onto a multidimensional hypercube $D = \{\boldsymbol{\theta} \in [a, b]^N\} \subset \mathbb{R}^N$:

$$\{h(x) : 0 \leq x \leq 1\} = \{\boldsymbol{\theta} \in \mathbb{R}^N : a \leq \theta_i \leq b, i \in N.\} \tag{1}$$

The function h is surjective; so for every point in the hypercube, there exists at least one point in the interval which maps onto it.

The first examples of space-filling curves date back to Peano [14]; the one we adopt here is due to Hilbert [15]. Their definition is given in the limit of infinitely many refinements of recursive constructions. Each recursive step discretises the space at a fixed resolution determined by a parameter m, thereby producing an m-approximation of the space. Specifically, the hypercube D is subdivided into $2^{N\times m}$ smaller hypercubes with 2^m subdivisions along each dimension. The Hilbert curve for an m-approximation, denoted $h_{N,m}$, traverses these unit hypercubes in a continuous manner, thus preserving spatial locality. As $m \to \infty$, the approximation converges to the true space-filling Hilbert curve, which fully covers the entire hypercube in the limit. An example of Hilbert curves $h_{N=3,m=3}(\cdot)$ can be seen on the left of Figure 1.

A property of Hilbert curves is that the multi-dimensional minimisation problem of a Lipschitz continuous function $f : \mathbb{R}^N \to \mathbb{R}^c$ ($N, c \in \mathbb{R}$) can be accurately reduced to the one-dimensional problem along the m-approximation of the Hilbert curve $h_{N,m}$ [16]:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^N} f(\boldsymbol{\theta}) = \min_{x\in[0,1]} f(h(x)) \approx \min_{x\in[0,1]} f(h_{N,m}(x)) = \min \tilde{f}(x); \tag{2}$$

where for brevity $\tilde{f}(x)$ denotes $f(h_{N,m}(x))$. Further, $\tilde{f}(x)$ is Hölder continuous with exponent $\alpha = 1/N$:

$$\forall x, x' \in [0,1] \colon |\tilde{f}(x) - \tilde{f}(x')| \leq H(|x - x'|)^{\frac{1}{N}}, \tag{3}$$

where $H = 2L\sqrt{N+3}$ is the Hölder constant and L is the Lipschitz constant of the original f.

Neural Networks with Lipschitz continuity. We consider Lipschitz continuous neural networks (NNs) $g : \mathbb{R}^N \to \mathbb{R}^c$. NNs comprising convolutional, fully connected, and contrast normalisation layers with ReLU activation functions are Lipschitz continuous [17]. Further, softmax layers, as well as sigmoid and hyperbolic tangent activation functions, also satisfy Lipschitz continuity [18]. We here focus on classification tasks where each input $\boldsymbol{x} \in \mathbb{R}^N$ is assigned to the class $\hat{y}$ among a set of classes $\{1, \ldots, c\}$ determined by the largest NN output, *i.e.*, $\hat{y} = \arg\max_{j=1,\ldots,c} g(x)_j$.

Local robustness verification. Given a NN $g : \mathbb{R}^N \to \mathbb{R}^c$, an input $\boldsymbol{x}$ to g, and a perturbation space $\Omega(\boldsymbol{x})$ of $\boldsymbol{x}$, the robustness verification problem establishes whether the class prediction of the network is consistent within the perturbation space. In other words, the problem is to determine whether:

$$\forall \boldsymbol{x'} \in \Omega(\boldsymbol{x}) \colon \arg\max_i g(\boldsymbol{x})_i = \arg\max_i g(\boldsymbol{x'})_i. \tag{4}$$

By taking $f(g, \boldsymbol{x}, \boldsymbol{x'}) = g(\boldsymbol{x'})_y - \max_{i\neq y} g(\boldsymbol{x'})_i$, where $y = \arg\max_i g(\boldsymbol{x})_i$, this is equivalent to establishing whether:

$$\forall \boldsymbol{x'} \in \Omega(\boldsymbol{x}) \colon f(g, \boldsymbol{x}, \boldsymbol{x'}) > 0. \tag{5}$$

A NN g is said to be certifiably robust on input $\boldsymbol{x}$ with respect to the perturbation space $\Omega(\boldsymbol{x})$ if Eq. (5) holds. Any violation of this property, *i.e.*, $\exists \boldsymbol{x'} \in \Omega(\boldsymbol{x}) \colon f(g, \boldsymbol{x}, \boldsymbol{x'}) < 0$, indicates the presence of a counterexample. A common perturbation space is the one generated by ℓ_p norms around $\boldsymbol{x}$, defined as $\Omega(\boldsymbol{x}) = \{\boldsymbol{x'} \colon ||\boldsymbol{x} - \boldsymbol{x'}||_p \leq \epsilon\}$ for a perturbation budget $\epsilon \in \mathbb{R}$.

Local geometric robustness. A perturbation space that is of particular interest in computer vision is defined in terms of geometric perturbations on the input, such as rotation, translation, scaling or combinations thereof [19]. A geometric perturbation is a 2D affine transformation $\boldsymbol{A_\theta}$ that provides a mapping between source coordinates (x^s, y^s) of the input and target coordinates (x^t, y^t) of the transformed input:

$$\begin{bmatrix} x^s \\ y^s \end{bmatrix} = \boldsymbol{A_\theta} \begin{bmatrix} x^t \\ y^t \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda\cos\gamma & -\sin\gamma & t^{\text{hor}} \\ \sin\gamma & \lambda\cos\gamma & t^{\text{ver}} \end{bmatrix} \begin{bmatrix} x^t \\ y^t \\ 1 \end{bmatrix}, \tag{6}$$

where $\boldsymbol{\theta} = [\gamma, \lambda, t^{\text{hor}}, t^{\text{ver}}]$ are the transformation parameters, with γ representing the rotation angle, λ denoting the scaling factor, and t^{hor}, t^{ver} controlling the horizontal and vertical translation. Each pixel value V_{x^y, y^t} in the transformed image can be computed by calculating the pre-image of the pixel under $\boldsymbol{A_\theta}$ and interpolating the (possibly non-integer) resulting coordinates using any interpolation scheme. We here adopt Spatial Transformation Networks [20] with bilinear interpolation to determine

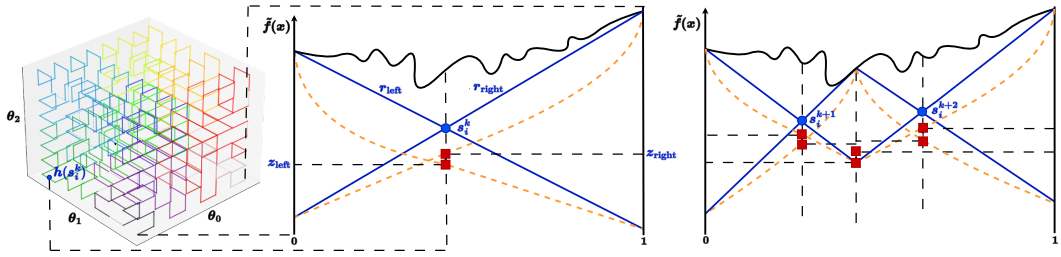

Figure 1: From left to right: Hilbert curve mapping; k-th iteration of HOVER's optimisation; The subsequent iterations of HOVER's optimisation.

these values: $V_{x^t,y^t} = \sum_n^H \sum_m^W U_{nm} \max(0, 1 - |x^s - m|) \max(0, 1 - |y^s - n|)$, where U_{hw} is the value of the pixel with coordinates (n, m).

Given interval constraints $\boldsymbol{\Theta} \subset \mathbb{R}^4$, the geometric perturbation space $\Omega(\boldsymbol{x}) = \{V(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ for an input $\boldsymbol{x}$ is the set of all transformed inputs for each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where each transformed input $V(\boldsymbol{x}, \boldsymbol{\theta})$ is obtained by determining $V_{x,y}$ for each pixel (x, y) of the input. Establishing the local robustness of NNs with respect to this space can be used to assess their robustness to geometric distortion effects, such as tilted camera orientation (rotation), positional shifts (translation), and zoom variations (scaling) [21].

The space has two important properties that enable the derivation and efficacy of the verification procedure introduced in the next section. First, prepending geometric transformation modules to Lipschitz continuous NNs preserves Lipschitz continuity, as proved in [22]. Second, the perturbation dimensionality of these modules is very low (the number of parameters) when compared to the input dimensionality of norm-based perturbation modules (the number of pixels). We will exploit this to provide an effective reduction to one dimension in a method that enables for the first time scalable global optimisation-based verification for more than one input dimensions.

We hereafter consider the local geometric robustness problem $\forall \boldsymbol{\theta} \in \boldsymbol{\Theta} : f(g', \boldsymbol{x}, V(\boldsymbol{x}, \boldsymbol{\theta})) > 0$, where g' denotes a NN prepended with a geometric module on input $\boldsymbol{x}$ [19]. Since g' and $\boldsymbol{x}$ are fixed, we briefly denote the problem by $\forall \boldsymbol{\theta} \in \boldsymbol{\Theta} : f(\boldsymbol{\theta}) > 0$.

3 Hölder-based Global Optimisation for Neural Network Verification

This section puts forward HOVER, a Hölder-based optimisation method for solving the robustness verification problem defined in the previous section. The method aims to find a solution to the optimisation problem $\min f(\boldsymbol{\theta})$ s.t. $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ is an N-dimensional hyperrectangle encoding the perturbation space around an input. If the solution to the problem is greater than zero, then the verification problem can be answered positively. To enable the utilisation of scalable 1D methods, HOVER first transforms the multivariate function of the optimisation problem into a univariate equivalent using the Hilbert space filling-curve. Some optimisation methods for the resulting univariate problem require knowledge of the Hölder constant [23], while some others do not [11]. Following the intractability of the accurate estimate of the constant [9], we here adapt a method from the latter category that relies on adaptive estimations of the constant throughout the optimisation process [9, 10, 18]. As we discuss in more detail below, while the method does not theoretically guarantee the identification of the global minimisers of the optimisation objective, it scores higher on soundness than leading, theoretically sound verification frameworks, while exhibiting significantly higher scalability. This is enabled by common optimisation practices that we employ and this section outlines.

In the following, we provide a technical exposition of HOVER. To ease its presentation, and without loss of generality, we assume that the input domain $\boldsymbol{\Theta}$ has been normalised to the hypercube $[0, 1]^N$. We begin with a short technical overview.

Overview. Figure 1 illustrates the first two iterations of HOVER. Having transformed the multivariate objective function into a univariate one, the algorithm iteratively operates on increasingly tighter intervals of the (one-dimensional) input range. For each interval, it computes a low-bounding

piecewise function (the orange/dashed lines in the figure) using an estimation of the Hölder constant for that interval. Based on this low-bounding function, it then computes a lower bound for the interval (the minimum between z_{left} and z_{right} in the figure). At each iteration, the algorithm chooses the interval with the lowest bound to split into tighter intervals at the point where the estimated lower bound is observed. Lower-bounding functions for the new sub-intervals are then computed to facilitate the next iteration. The algorithm terminates when an optimisation budget ϵ is reached that reflects the minimum length of the selected interval. Below, we discuss this procedure in detail.

Initialisation. HOVER is initialised by: (i) transforming the multi-dimensional optimisation problem $\min f(\boldsymbol{\theta})$ s.t. $\boldsymbol{\theta} \in [0,1]^N$ to the one-dimensional problem $\min \tilde{f}(x)$ s.t. $x \in [0,1]$ using the Hilbert space-filling curve, (ii) setting $\mathcal{I} = \{[0,1]\}$ to be the set of initial intervals, and (iii) letting $\mathcal{O} = \{\}$ to be the set of already considered intervals. Then, for each iteration $k \geq 1$, HOVER executes the following steps.

Step 1 (Adaptive estimation of the Hölder constant). For each interval $i \in \mathcal{I}$, with $i = [a,b]$, HOVER computes a *local* Hölder constant $H_i = \frac{|\tilde{f}(b) - \tilde{f}(a)|}{|a-b|^{1/N}}$, and a *global* Hölder constant $h_k = \max\{H_i \mid i \in \mathcal{I}\}$. Based on these constants, it derives its (adaptive) estimation of the Hölder constant for interval i as $\hat{H}_i = r \cdot \max\{\lambda, \gamma, \xi\}$, where:

- $\lambda = \max\{H_j \mid j = i \text{ or } j \text{ is adjacent to } i\}$ is the local component of the estimation that reflects the maximum constant between the local constants of interval i and its adjacent intervals (*i.e.*, intervals that have a common bound);
- $\gamma = h_k \frac{|a-b|}{X_{\max}}$ is the global component of the estimation, where $X_{\max} = \max\left\{(b'-a')^{1/N} \mid [a',b'] \in \mathcal{I}\right\}$ is the widest interval;
- ξ is a small value that prevents $\hat{H}_i$ from becoming 0, accounting for $\tilde{f}(x)$ varying over $[0,1]$;
- $r > 1$ is the reliability parameter of the algorithm.

Intuitively, the adaptive estimation $\hat{H}_i$ is dominated by the global component whenever an interval is large (and thus the local estimates are not reliable), and by the local component whenever an interval is small (and thus the local estimates are more accurate). As discussed in more detail below, the reliability parameter r mitigates potential underestimations of the constant.

Step 2 (Estimation of the lower bounds of the intervals). For each interval $i \in \mathcal{I}$, with $i = [a,b]$, the algorithm computes the point

$$s_i = \frac{b+a}{2} - \frac{\tilde{f}(b) - \tilde{f}(a)}{2\hat{H}_i (b-a)^{\frac{1-N}{N}}}. \tag{7}$$

This point is the intersection of the lines $r_{\text{left}}(x)$ and $r_{\text{right}}(x)$ (see the blue/solid lines in Figure 1), which are defined as

$$\begin{aligned} r_{\text{left}}(x) &= -\hat{H}_i (b-a)^{\frac{1-N}{N}} x + \hat{H}_i (b-a)^{\frac{1-N}{N}} a + \tilde{f}(a), \\ r_{\text{right}}(x) &= \hat{H}_i (b-a)^{\frac{1-N}{N}} x - \hat{H}_i (b-a)^{\frac{1-N}{N}} b + \tilde{f}(b). \end{aligned} \tag{8}$$

These lines relax the piecewise lower bounding functions of $\tilde{f}$ within the interval (the orange/dashed lines in the figure); we refer to [9] for a formal description of the functions using the estimated Hölder constant. The lower bound l_i of the interval is then estimated as $l_i = \min(z_{\text{left}}, z_{\text{right}})$, where

$$z_{\text{left}} = \tilde{f}(a) - \hat{H}_i (s_i - a)^{1/N}, \; z_{\text{right}} = \tilde{f}(b) - \hat{H}_i (b - s_i)^{1/N}. \tag{9}$$

The bound l_i corresponds to the minimum value of the lower bounding functions evaluated at the s_i.

Step 3 (Convergence and refinement). HOVER selects the interval $i \in \mathcal{I}$ with the minimum lower bound estimate l_i. Then,

- If the length of the interval $i = [a,b]$ is smaller than the optimisation budget, *i.e.*, $|b-a| \leq \epsilon$, it executes to **Step 4** and terminates;
- Otherwise, it splits the selected interval with respect to $s_i = [a',b']$, and updates $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\} \cup \{[a,a'],[a',b]\}$, $\mathcal{O} \leftarrow \mathcal{O} \cup \{i\}$. It then repeats from **Step 1**.

Step 4 (Calibration and output). HOVER computes an estimation of the minimum of the function as $\tilde{f}_m = \min\left\{\tilde{f}(a), \tilde{f}(b) \mid [a,b] \in \mathcal{I}\right\}$, and an estimation of the lower bound of the function as

$l_m = \min\{l_j \mid j \in \mathcal{I} \cup \mathcal{O}\}$. l_m is then calibrated as $l_m \leftarrow l_m - \eta$, where $\eta = L \cdot 2^{-(m+1)}\sqrt{N} + H \cdot (\epsilon/2)^{1/N}$, L and H being the latest estimates of the global Lipschitz and Hölder constants, and m is the resolution of the Hilbert approximation. The calibration, which is theoretically analysed below, accounts for (i) approximation errors in the dimensionality reduction along the Hilbert curve, and (ii) the constrained nature of the optimisation budget ϵ with which the algorithm operates. Following the calibration, HOVER produces its output as follows:

- If $l_m > 0$, then it returns *robust*, *i.e.*, a positive answer to the robustness of the underlying network.
- If $\tilde{f}_m < 0$, then it returns *non-robust*, along with a counterexample $h_{N,m}(x)$ corresponding to the value for which $\tilde{f}(x) = \tilde{f}_m$.

We now proceed to analyse the algorithm's soundness and examine practical methods for sustaining high reliability and computational efficiency. We begin by showing that a calibrated (as per **Step 4**) lower bound for the reduced one-dimensional space translates to a lower bound for the original N-dimensional space.

Theorem 1. *Let $l_h^\star$ be a lower bound of the one-dimensional problem $\min \tilde{f}(x)$ s.t. $x \in [0,1]$ over the Hilbert space-filling curve. Then we have that*

$$l_h^\star - L \cdot 2^{-(m+1)}\sqrt{N} - H \cdot (\epsilon/2)^{1/N} \leq l^*, \tag{10}$$

where l^* is the optimal solution of the multi-dimensional problem $\min f(\boldsymbol{\theta})$ s.t. $\boldsymbol{\theta} \in [0,1]^N$.

Proof. The proof is included in the Appendix. □

Note that the fist calibration term results from the approximation of the Hilbert curve reduction, while the second is a consequence of the limited optimisation budget. When the resolution of the Hilbert approximation is high enough, *e.g.,* $m = 50$ in our experiments, the magnitude of the former term is negligible. Differently, the magnitude of the latter term grows with the number of dimensions, thus hindering the efficacy of HOVER to high-dimensional input domains.

Next, we show that given a sufficiently enough large value for the reliability parameter r, HOVER implements a sound verification procedure.

Theorem 2. *There exists r^* s.t. for all $r > r^*$,* HOVER *outputs* robust *iff* $\forall \boldsymbol{\theta} \in [0,1]^N : f(\boldsymbol{\theta}) > 0$.

Proof. The result follows immediately from Theorem 1 and Theorem 3.8 in [9]. □

Note that Theorem 2 does not provide a constructive way of determining r^*; we conjecture that such a procedure may have high complexity. Consequently, in practice, the Hölder constant can be underestimated at any iteration and interval, which may impact the localisation of the global minimisers and the convergence speed. Note that without knowing the true Lipschitz/Hölder constant, this also applies to existing global optimisation–based verification methods [18, 24], and overlooking it may lead to potentially unsound results, even if their algorithm converges, it could be a local minimum rather than the global one. In the light of this, we below discuss operational enhancements that remedy both potential pitfalls.

Practical enhancements. To ensure high reliability, following convergence (*i.e.*, when the optimisation budget is reached), HOVER iteratively increases the value of the reliability parameter r until either (i) a different interval is selected at **Step 3**, or (ii) a time limit (given as a parameter) is reached. Intuitively, if the algorithm converges to a local minimum following an underestimation of the Hölder constant, the iterative adjustment of the reliability parameter will eventually trigger an escape from said minimum. To further enable high practical efficacy, HOVER implements two strategies. First, it employs a heuristic whereby it dynamically adapts the Hölder constant based on both local and global information as detailed in **Step 1**. Second, for every iteration, following the selection of an interval $i = [a, b]$ and division thereof as per split point s_i at **Step 3**, it re-estimates the lower bound of an interval j at the next iteration only if one of the following conditions hold: (i) j is adjacent or contained in i; (ii) the length of i is equal to $X_{\max}$; (iii) the local Hölder constant for the subintervals of i is greater than the global Hölder constant h_k. These express the necessary conditions for triggering a change in the estimation of the lower bound l_j of each interval j (as per

Table 1: Evaluation results on 500 images from ImageNet against the perturbation combination of rotation, translation and scaling. Baselines performance is adopted from [22].

Perturbation	Model	Clean Acc	NN Params	Adversarial Accuracy (%)		Certified Accuracy (%)	
				GeoRobust	HOVER	GeoRobust	HOVER
R(20°) + T(10%) + S(10%)	Inception V3	73.6	24M	28.2	24.4	24.2	23.8
	ResNet34	72.0	22M	-	27.4	-	25.8
	ResNet50	78.4	26M	54.0	42.6	31.1	40.6
	ResNet101	80.0	45M	54.2	49.4	48.2	48.0
	ResNet152	79.4	60M	53.8	49.0	46.2	48.4
	Mixer	72.2	60M	27.2	24.8	23.4	24.2
	Gmlp	78.0	19M	40.8	37.6	36.8	34.2
	Swin	80.2	88M	34.6	22.2	13.2	10.0
	Large $\text{Vit}_{16\times16}$	83.4	300M	49.2	42.2	40.2	34.8

the definition of the adaptive estimation of the Hölder constant $\hat{H}_j$ in **Step 1**). Taken together these further contribute towards achieving high efficiency and a high degree of correctly computed results.

4 Experimental Evaluation

Experimental Setup. Our experiments were conducted on a machine equipped with an Intel i7-12700K CPU with 78GB RAM running kernel 5.15 and an RTX 3090 Ti GPU with 24GB of graphics memory. All the code is written using PyTorch, and the Hilbert space-filling curve is implemented with the hilbertcurve library [25]. Our experimental evaluation is aimed to evaluate the practical applicability of the approach. We establish this by assessing the scalability of the approach on very large NNs and its practical reliability. As we discuss below, our findings suggest that the method scales to models such as vision transformers and video models that could not be verified before, and the implementation achieves the highest level of correctly answered verification queries.

In terms of geometric perturbations, we denote $\text{R}(\gamma)$ as the rotation operation, where the angle varies within the range $\pm\gamma$, and $\text{S}(\lambda)$ as the scaling operation, where the scaling factor ranges between $1 \pm \lambda$. Let $\text{T}(t)$ represent the translation operation, shifting an input by up to $\pm t$ proportionally in both the horizontal and vertical directions. Here we consider the combination of these three types of geometric transformations to evaluate the model's robustness, both in terms of its adversarial accuracy, and its certified accuracy. The first measures the percentage of test inputs for which a counter-example was not found by the method. The second measures the percentage of samples reported to be robust in the geometric neighbourhood considered. Whenever a method can solve all queries the two measures are identical. We report only the highlights in the rest of this section but base our conclusions on the comprehensive benchmarking for the method also reported in the Appendix.

Large NNs for Image Classification. To evaluate the performance of HOVER on image classification for large NNs, we benchmarked the adversarial accuracy and certified accuracy obtained by the tool on 9 Models of different sizes, ranging from 19M (Gmlp) to 300M (Large $\text{ViT}_{16\times16}$) tuneable parameters including several ResNet models, up to ResNet152, normally trained to a good level of accuracy. The dataset used is ImageNet on inputs of $3 \times 224 \times 224$. The verification queries consisted of any input transformation with rotation, translation and scaling with large parameters (20°, 10%, and 10%, respectively). To our knowledge, GeoRobust [22] is the only available tool that can handle such queries on high dimensional inputs for such large NNs. In particular, none of the tools in VNN-COMP [4], nor [21] can resolve such queries.

Table 1 reports the results obtained. From the results we find that HOVER significantly outperforms GeoRobust both in terms of adversarial and certified accuracy, and with a smaller number of undecided cases. GeoRobust is only shown to have superior performance on Large $\text{Vit}_{16\times16}$, Gmlp and Swin. However, further analysis of these results indicate that GeoRobust often incorrectly concludes safety. Indeed, HOVER identified several counterexamples (6 for ResNet101, 7 for ResNet152, 18 for Swin) to verification queries that were reported *robust* by GeoRobust. We suspect this is because, without a sufficient number of iterations, GeoRobust's underlying global optmisation can often use underestimations of the Lipschitz constant; this is a behaviour not discussed in [22].

The robustness results suggest that the ViT model is less robust than some ResNet models. This raises the question of why the patch-based attention mechanisms do not translate into improved robustness [26]. The results here only refer to geometric robustness and require further analysis.

Large NNs for Video Classification. To further evaluate the performance of HOVER we here report the results of the experiments that were ran to assess the robustness of large models used for video classification. For this we considered end-to-end RGB 3D-NNs without flow information trained on the Kinetics-400 dataset [27]. Specifically, we evaluated 6 pre-trained NNs from the open-source library PyTorchVideo [28], with network parameters ranging from 3.79M to 32.45M, and up to $32 \times 3 \times 256 \times 256$ input dimensions: Slow-R50 [29], R(2+1)D-R50 [30], X3D_M [31], I3D-R50 [32], CSN-R101 [33], and C2D-R50 [34].

For the evaluation, we randomly selected 200 videos from the dataset and evaluated the robustness of the models against perturbations applied to entire clips and single frames (see the Appendix for technical details). The perturbations consisted of combinations of rotation, scaling and translation, using the same perturbation intensity used for the images above.

Table 2: Benchmarking static geometric robustness of video classification models against geometric transforms (R(20°)+S(10%) + T(10%)).

Model	Frame Length	Frame Rate	Params (M)	Clean Acc (%)	Adversarial Acc (%)	Certified Acc (%)
X3D_M	16	5	3.79	78.0	37.0	36.0
CSN-R101	32	2	22.21	79.0	47.0	44.0
C2D-R50	8	8	24.33	73.5	39.0	37.5
I3D-R50	8	8	28.04	73.0	42.5	42.0
R(2+1)D-R50	16	5	28.11	76.0	45.0	45.0
Slow-R50	8	8	32.45	76.5	44.0	43.0

To the best of our best knowledge, the only two verification methods applicable to video tasks are [35] and [36]. However, the methods can only scale to small NNs, hence they are not comparable to HOVER's capabilities.

The results are reported in Table 2. It can be observed that HOVER was able to resolve a large proportion of the verification queries with a minimum rate of undecided cases. To our knowledge, this is the first time that large video classifiers are evaluated for local robustness. The results point to CSN-R101 and R(2+1)D-R50, which achieved the highest adversarial accuracy (47.0% and 45.0%) and certified accuracy (44.0% and 45.0%). These findings indicate that architectural refinements and expanding model capacity could potentially benefit robustness against geometric transformations.

Table 3: Soundness validation on SoundnessBench. Baseline performances are adopted from [6].

Benchmark	Input Dim	Tool	Verified	Falsified	Unknown	Unsound
CNN1	25-75	$\alpha\beta$-CROWN	18	0	58	0
		PyRAT	10	0	66	0
		HOVER	25	0	51	0
CNN2	25-75	$\alpha\beta$-CROWN	13	0	61	0
		PyRAT	5	0	69	0
		HOVER	17	0	57	0
CNN3	25-75	$\alpha\beta$-CROWN	7	1	68	0
		PyRAT	5	0	71	0
		HOVER	11	0	65	0
CNN AvgPool	25-75	$\alpha\beta$-CROWN	30	7	13	10
		PyRAT	0	0	60	0
		HOVER	16	0	44	0
CNN Tanh	25-75	$\alpha\beta$-CROWN	0	0	38	0
		PyRAT	0	0	38	0
		HOVER	20	0	18	0
CNN Sigmoid	25-75	$\alpha\beta$-CROWN	2	0	29	0
		PyRAT	2	0	29	0
		HOVER	19	2	10	0
MLP	10	$\alpha\beta$-CROWN	20	2	49	0
		PyRAT	12	0	59	0
		HOVER	26	1	44	0

Soundness Validation. We discussed in Sections 1 and 3 that the proposed verification method based on global optimisation may return unsound results. As discussed, this theoretical possibility may be mitigated, as it is routinely done in optimisation, by careful the choice of the optimisation parameters.

In what follows we evaluate the empirical soundness of HOVER. We do this in two ways. Firstly, we evaluate the results obtained by the tool on SoundnessBench [6]. This is a recently released neural network verification benchmark, which was designed for validating the correctness of verifiers by including the ground truth of the verification queries. Secondly, we report the results obtained by the tool against low-dimensionality perturbations from VNN-COMP [4]. In this case the ground truth is not known, but like VNN-COMP, we compare the results against those produced by SoA tools, by taking the agreement among all verifiers as ground truths. In total, we evaluate the soundness of HOVER about 500 results. All results produced by HOVER on SoundnessBench were correct (in line with the ground truth); all results produced by HOVER on the VNN-COMP tests were in line with those reported by the $\alpha\beta$-CROWN [37].

Table 4: Evaluation on two VNN-COMP benchmarks. Baseline performances are adopted from [4].

Benchmark	#Params	Tool	Verified	Falsified	Unknown	Unsound
CCTSDB Yolo	100K	$\alpha\beta$-CROWN	11	28	0	0
		PyRAT	0	2	37	0
		HOVER	11	28	0	0
TLL Verify Bench	17k-67M	$\alpha\beta$-CROWN	15	17	0	0
		PyRAT	15	17	0	0
		NeuralSAT	15	0	0	17
		HOVER	15	17	0	0

We report the results obtained on SoundnessBench on Table 3. The benchmark comprises 24 models (primarily CNNs and MLPs) and includes 240 verification queries that ought to be resolved as *robust* and 186 that ought to be resolved as *non-robust*. Note that the latter include carefully hidden adversarial examples that are challenging for verifiers to discover. The performance of two SoA verifiers, $\alpha\beta$-CROWN [37] and pyRAT [38], are also reported for comparison. As shown, HOVER returned the correct result for all the queries. Differently, $\alpha\beta$-CROWN, which is regarded as SoA, reported several incorrect results. We refer to the Appendix for detailed performance of each method.

Table 4 reports the results obtained on the CCTSDB Yolo and TLL Verify Bench benchmarks from VNN-COMP [4]. The benchmarks were selected because of the low-input dimensionality ($N = 2$) of the perturbations they consider, thus falling within the scope of HOVER. We observe that HOVER achieves the same performance as $\alpha\beta$-CROWN, without exhibiting any unknown or unsound cases. Differently, another two SoA verifiers, PyRAT [38] and NeuralSAT [39], either fail to prove robustness for several cases or report incorrect results.

5 Related work

An extensive body of literature exists on the verification of NNs against ℓ_p-bounded and other perturbations; we refer to [2, 40] for a survey on the area. A variety of methods are used, from Mixed-Integer Linear Programming [41–43], to SMT [39, 44], branch-and-bound [45–48], abstract interpretation [38, 49–51], symbolic interval propagation methods [52–54]. Some of these methods, notably symbolic interval propagation, outperform the present approach for large dimensionality problems. However, as shown in the previous section, HOVER considerably outperforms all SoA on geometric perturbations, including the approaches targetting geometric robustness directly [21, 22, 55–57] (see also a discussion in the Appendix). Unlike the approach here presented, many of the cited methods are shown to be theoretically sound. Yet, in practice, their theoretical soundness often translates to unsound implementations, *e.g.*, floating point approximations [5] may impact the soundness of the bounds generated by symbolic interval propagation methods. In turn, this results in errors in actual verification experiments as recently confirmed in [6]. In Section 4, we empirically validated HOVER and found not a single error in actual verification experiments; in contrast, we found that some of the SoA tools, while theoretically sound, answered some queries incorrectly.

Much closer to HOVER are existing methods based on global optimisation [18, 24, 58]. The key difference between HOVER and these methods is that the former couples Hölder optimisation with a dimensionality reduction technique, thereby scaling to larger models and to higher dimensions, as we empirically demonstrated. We note that existing optimisaton methods provably converge only if particular (reliability) parameters can be chosen. However, this choice is closely related to providing an upper bound of the Lipschitz constant. This is normally intractable for large models, for which only estimations of the constant can be used in practice, thereby potentially resulting in unsound results, as we empirically observed. In contrast, no incorrect results were produced by HOVER during our extensive validation (see previous section).

6 Conclusions

We presented HOVER, a novel method for the verification of geometric robustness of NNs based on Hölder optimisation and dimensionality reduction. We demonstrated that HOVER outperforms SoA methods on vision classification models up to millions of tuneable parameters and large inputs, and also enables the verification of large video classifiers for the first time. This makes it possible, we believe for the first time, to apply verification methods to the validation in safety critical applications in vision. While we do not provide a constructive proof of convergence, we validated the soundness empirically by assessing the results on hundreds of queries. We observed that a robustness analysis concerns the aggregation of several results and it is therefore not skewed by a very low error rate.

In terms of limitations of HOVER, the present SoA, notably symbolic interval propagation, scales better on large dimensionality problems such as noise perturbations; so they remain the approach of choice for those problems. In further work, we intend to focus on implementing HOVER in other problems with low dimensionality such as contrast, luminosity, bias field [59] and blur [60] as well as on extending the dimensionality reduction here devised to scale to more dimensions.

References

[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[2] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.

[3] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

[4] Christopher Brix, Stanley Bak, Taylor T Johnson, and Haoze Wu. The fifth international verification of neural networks competition (vnn-comp 2024): Summary and results. *arXiv preprint arXiv:2412.19985*, 2024.

[5] Kai Jia and Martin Rinard. Exploiting verified neural networks via floating point numerical error. In *Static Analysis: 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17–19, 2021, Proceedings 28*, pages 191–205. Springer, 2021.

[6] Xingjian Zhou, Hongji Xu, Andy Xu, Zhouxing Shi, Cho-Jui Hsieh, and Huan Zhang. Testing neural network verifiers: A soundness benchmark with hidden counterexamples. *arXiv preprint arXiv:2412.03154*, 2024.

[7] Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196, 2021.

[8] Alex Serban, Erik Poll, and Joost Visser. Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 53(3):1–38, 2020.

[9] Daniela Lera and Ya D Sergeyev. Global minimization algorithms for holder functions. *BIT Numerical mathematics*, 42:119–133, 2002.

[10] Daniela Lera and Ya D Sergeyev. Lipschitz and hölder global optimization using space-filling curves. *Applied Numerical Mathematics*, 60(1-2):115–129, 2010.

[11] Yaroslav D Sergeyev, Roman G Strongin, and Daniela Lera. *Introduction to global optimization exploiting space-filling curves*. Springer Science & Business Media, 2013.

[12] Vladimir Grishagin, Ruslan Israfilov, and Yaroslav Sergeyev. Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. *Applied Mathematics and Computation*, 318:270–280, 2018.

[13] Houshang H Sohrab. *Basic real analysis*, volume 231. Springer, 2003.

[14] G Peano. Sur une courde qui remplit touteune aire plane. *Math. Ann.*, 36, 1890.

[15] David Hubert. Ueber die stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[16] Roman G Strongin and Yaroslav D Sergeyev. *Global optimization with non-convex constraints: Sequential and parallel algorithms*, volume 45. Springer Science & Business Media, 2013.

[17] C Szegedy. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[18] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2651–2659, 2018.

[19] Panagiotis Kouvaros and Alessio Lomuscio. Formal verification of cnn-based perception systems. *arXiv preprint arXiv:1811.11373*, 2018.

[20] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 2015.

[21] Ben Batten, Yang Zheng, Alessandro De Palma, Panagiotis Kouvaros, and Alessio Lomuscio. Verification of geometric robustness of neural networks via piecewise linear approximation and lipschitz optimisation. In *ECAI 2024*, pages 2362–2369. IOS Press, 2024.

[22] Fu Wang, Peipei Xu, Wenjie Ruan, and Xiaowei Huang. Towards verifying the geometric robustness of large-scale neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15197–15205, 2023.

[23] Eric Gourdin, Brigitte Jaumard, and Rachid Ellaia. Global optimization of hölder functions. *Journal of Global Optimization*, 8:323–348, 1996.

[24] Chi Zhang, Zhen Chen, Peipei Xu, Geyong Min, and Wenjie Ruan. Verification on out-of-distribution detectors under natural perturbations. *Machine Learning*, 114(3):77, 2025.

[25] Gabriel Altay. Hilbertcurve.

[26] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. In *International conference on machine learning*, pages 27378–27394. PMLR, 2022.

[27] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[28] Haoqi Fan, Tullie Murrell, Heng Wang, Kalyan Vasudev Alwala, Yanghao Li, Yilei Li, Bo Xiong, Nikhila Ravi, Meng Li, Haichuan Yang, et al. Pytorchvideo: A deep learning library for video understanding. In *Proceedings of the 29th ACM international conference on multimedia*, pages 3783–3786, 2021.

[29] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 3154–3160, 2017.

[30] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[31] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020.

[32] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[33] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5552–5561, 2019.

[34] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[35] Min Wu and Marta Kwiatkowska. Robustness guarantees for deep neural networks on videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 311–320, 2020.

[36] Samuel Sasaki, Diego Manzanas Lopez, Preston K Robinette, and Taylor T Johnson. Robustness verification of video classification neural networks.

[37] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in neural information processing systems*, 35:1656–1670, 2022.

[38] Augustin Lemesle, Julien Lehmann, and Tristan Le Gall. Neural network verification with pyrat. *arXiv preprint arXiv:2410.23903*, 2024.

[39] Hai Duong, ThanhVu Nguyen, and Matthew Dwyer. A dpll (t) framework for verifying deep neural networks. *arXiv preprint arXiv:2307.10266*, 2023.

[40] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, Mykel J Kochenderfer, et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, 4(3-4):244–404, 2021.

[41] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[42] Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

[43] Panagiotis Kouvaros and Alessio Lomuscio. Towards scalable complete verification of relu neural networks via dependency-based branching. In *IJCAI*, pages 2643–2650, 2021.

[44] Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, et al. Marabou 2.0: a versatile formal analyzer of neural networks. In *International Conference on Computer Aided Verification*, pages 249–264. Springer, 2024.

[45] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021.

[46] Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020.

[47] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

[48] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

[49] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2018.

[50] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. *Advances in neural information processing systems*, 31, 2018.

[51] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

[52] Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio, and Ruth Misener. Efficient verification of relu-based neural networks via dependency analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3291–3299, 2020.

[53] Patrick Henriksen and Alessio Lomuscio. Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In *IJCAI*, pages 2549–2555, 2021.

[54] Panagiotis Kouvaros, Benedikt Brückner, Patrick Henriksen, and Alessio Lomuscio. Dynamic back-substitution in bound-propagation-based neural network verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27383–27391, 2025.

[55] Linyi Li, Maurice Weber, Xiaojun Xu, Luka Rimanic, Bhavya Kailkhura, Tao Xie, Ce Zhang, and Bo Li. Tss: Transformation-specific smoothing for robustness certification. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 535–557, 2021.

[56] Zhongkai Hao, Chengyang Ying, Yinpeng Dong, Hang Su, Jian Song, and Jun Zhu. Gsmooth: Certified robustness against semantic transformations via generalized randomized smoothing. In *International Conference on Machine Learning*, pages 8465–8483. PMLR, 2022.

[57] Mislav Balunovic, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin Vechev. Certifying geometric robustness of neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[58] Chi Zhang, Wenjie Ruan, and Peipei Xu. Reachability analysis of neural network control systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15287–15295, 2023.

[59] Patrick Henriksen and Alessio Lomuscio. Robust training of neural networks against bias field perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14865–14873, 2023.

[60] Benedikt Brückner and Alessio Lomuscio. Verification of neural networks against convolutional perturbations via parameterised kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27215–27223, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction have clearly stated the main contributions, scope and potential limitation.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation and potentially unsound results may be obtained via our algorithm, but this seldom happens.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The complete proofs are provided in the appendix. Theorems are properly referenced in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe the proposed algorithm in detail and include its pseudocode in the Appendix. An experienced researcher should be able to reproduce the method within a reasonable time. The code will be open-sourced.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code will be open-sourced, and the datasets we evaluate are publicly available; no private benchmarks are used.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed settings for our experiments are provided in the experiments section and in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our method is deterministic, producing identical results when executed on the same machine.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed settings, environment configurations, and software package information for our experiments are provided in the experiments section and in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read and acknowledged the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Societal impacts have been discussed explicitly in the introduction. As the paper concerns a technical approach to AI Safety. Specifically, it is intended to contribute towards the assessment of AI systems before they are deployed. We regard this as a strongly positive social impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have properly cited the original paper that produced the code package or dataset.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.