Scalable Neural Network Geometric Robustness Validation via Hölder Optimisation

Yanghao Zhang[1,2] **Panagiotis Kouvaros**[1] **Alessio Lomuscio**[1,2]
[1] Safe Intelligence, UK
[2] Department of Computing, Imperial College London, UK
{yanghao, panagiotis, alessio}@safeintelligence.ai

Abstract

Neural Network (NN) verification methods provide local robustness guarantees for a NN in the dense perturbation space of an input. In this paper we introduce $\mathrm{H}^2\mathrm{V}$, a method for the validation of local robustness of NNs against geometric perturbations. $\mathrm{H}^2\mathrm{V}$ uniquely employs a Hilbert space-filling construction to recast multi-dimensional problems into single-dimensional ones and Hölder optimisation, iteratively refining the estimation of the Hölder constant for constructing the lower bound. In common with methods, Hölder optimisation might theoretically converge to a local minimum, thereby resulting in a robustness result being incorrect. However, we here identify conditions for $\mathrm{H}^2\mathrm{V}$ to be provably sound, and show experimentally that even outside the soundness conditions, the risk of incorrect results can be minimised by introducing appropriate heuristics in the global optimisation procedure. Indeed, we found no incorrect results validated by $\mathrm{H}^2\mathrm{V}$ on a large set of benchmarks from SoundnessBench and VNN-COMP. To assess the scalability of the approach, we report the results obtained on large NNs ranging from Resnet34 to Resnet152 and vision transformers. These point to SoA scalability of the approach when validating 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 the robustness validation of 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 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, and illumination variations. Evaluating the robustness of a model, *i.e.*, its resistance to such small input changes, 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. With the exceptions discussed in the Related Work (Section 5), methods for the verification of local robustness are often theoretically *sound* (if a method reports that the model is either locally robust or not robus in a region, then that is guaranteed to be the case), and often *complete* (given unlimited time and resources a method can always resolve the local robustness query).

A well-known difficulty of sound and complete verification methods is their scalability: the robustness verification problem is theoretically NP-hard [3] and present SoA methods often fail to scale to large models, large inputs, or large perturbations [4], thereby hindering the application of the methods

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

in many noteworthy applications. Consequently, incomplete methods like adversarial testing are routinely used in applications to evaluate the robustness of large models [5, 6]. Yet, adversarial testing is known to fail to identify fragilities in a large number of cases, potentially instilling a false sense of robustness in the developer.

In this paper, we leverage on the considerations above to introduce H^{2}V, a method based on **H**ilbert curve mapping and **H**ölder optimisation for robustness **V**alidation. Differently from some verification methods, H^{2}V is theoretically unsound in general. The practical impact of this is less significant than what may initially appear for two reasons. Firstly, in applications the assessment of a model's robustness is typically based on the analysis of a large number of input samples and perturbations of varying magnitudes. It is the *aggregation of these results*, not any single query, that enables the analysis of a model's robustness and the comparison across models. Secondly, the actual correctness of individual verification queries in SoA verification tools is often hindered by inherent system-level floating point precision errors and other issues [7, 8], thereby rendering theoretical soundness guarantees less significant in practice.

H^{2}V is based on Hölder optimisation and follows advanced developments in the area of global optimisation [9–11]. As such, in line with many global optimisation methods [12] for NN analysis, its convergence to the global minimum can be assured theoretically only if appropriate optimisation parameters are chosen. Given this, we refer to H^{2}V as a *robustness validation method*, rather than one for robustness verification, because in general the method may be unsound. However, in what follows we identify circumstances where the method is theoretically sound, thereby falling into the category of traditional verification methods. In cases where this assumption cannot be established, we demonstrate that in practice the optimisation procedure at the heart of H^{2}V results in no incorrect robustness results in all the validation benchmarks that we studied, including SoundnessBench [8], indeed outperforming in terms of soundness all current SoA and theoretically sound methods.

A major feature of H^{2}V lies in its scalability. As we demonstrate below, H^{2}V enables the validation of models with hundreds of millions of tunable parameters, thereby enabling the robustness analysis of models in many present applications.

In summary, our contributions are as follows:

- We propose H^{2}V, a global optimisation method for the validation of NNs based on space-filling dimensionality reduction and Hölder optimisation. We provide theoretical conditions for theoretical convergence, hence soundness. We illustrate that when these theoretical convergence conditions are not met, the potential of a robustness error in a single query is well contained. Indeed, no errors were found in the extensive evaluation reported.
- We use H^{2}V to validate the local robustness of models of up to 300M tunable 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 H^{2}V to validate the geometric robustness of 3D ResNet models in video classification tasks for streams of $32 \times 3 \times 256 \times 256$ inputs.

The rest of the paper is organised as follows. In Section 2 we present key notions of use throughout. We present H^{2}V in Section 3 where we give the technical details of the validation approach and present soundness conditions. In Section 4, we evaluate H^{2}V 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 validation method presented in the next section.

Hölder/Lipschitz constant. A function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ 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 definitions are 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 (or attack, or fragility). 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, isotropic 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:

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

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

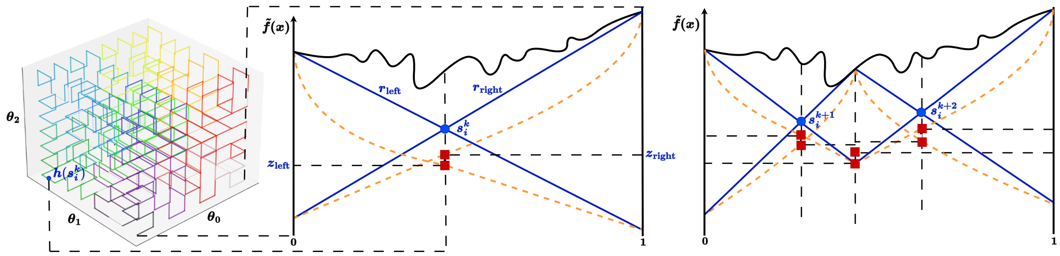

Figure 1: From left to right: Hilbert curve mapping; k-th iteration of H^{2}V's optimisation; The subsequent iterations of H^{2}V's optimisation.

pixel value V_{x^t,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 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 (isotropic scaling) [21].

Two key properties enable the derivation and efficiency of the validation procedure introduced in the next section. Firstly, prepending geometric transformation modules to Lipschitz continuous NNs preserves Lipschitz continuity [22]. Secondly, the perturbation dimensionality, *i.e.*, the number of parameters, of these modules is very low when compared to the input dimensionality, *i.e.*, the number of pixels, of norm-based perturbation modules. We will exploit this to provide an effective reduction to one dimension for a global optimisation method for more than one input dimensions.

In the following we 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 Validation

In this section we present H^{2}V, a Hölder-based global optimisation method for addressing the local geometric robustness 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 robustness problem is answered positively by H^{2}V. To enable the utilisation of scalable 1D methods, H^{2}V 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, as we demonstrate below, the potential for any potential unsound result is well-contained in view of widely employed settings in the resulting global optimisation problem.

In the following, we provide a technical exposition of H^{2}V. To ease its presentation, and without loss of generality, we assume that the input domain $\boldsymbol{\Theta}$ is the hypercube $[0, 1]^N$. We begin with a short technical overview.

Overview. Figure 1 illustrates two consecutive iterations of H^{2}V. 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 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. H^{2}V 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 $k \geq 1$, H^{2}V executes the following steps.

Step 1 (Adaptive estimation of the Hölder constant). For each interval $i \in \mathcal{I}$, with $i = [a,b]$, H^{2}V 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 $\hat{H}_i = r \cdot \max\{\kappa, \eta, \xi\}$, where:

- $\kappa = \max\{H_j \mid j = i \text{ or } j \text{ is adjacent to } i\}$ is the local component of the estimation, which represents the maximum value among the local constants of interval i and its neighbouring intervals (*i.e.*, all intervals within a given number of hops n_κ from i, including i itself.
- $\eta = h_k \frac{|a-b|}{X_{\max}}$ 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, thereby 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). A a practical enhancement is introduced in more details below to mitigate 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 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 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). H^{2}V 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 **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). H^{2}V 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 $l_m = \min\{l_j \mid j \in \mathcal{I} \cup \mathcal{O}\}$. The bound 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 are the present 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 within which the algorithm operates. Following the calibration, H^{2}V 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 first 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 H^{2}V to high-dimensional input domains.

Next, we show that given a sufficiently enough large value for the reliability parameter r, H^{2}V implements a sound verification procedure.

Theorem 2. *There exists r^* s.t. for all $r > r^*$,* H^{2}V *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^*. 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. Consequently, H^{2}V may output *robust* when the local geometric robustness problem is not robust. This reflects all existing global optimisation–based verification methods that do not necessitate knowledge of the true Lipschitz/Hölder constant [18, 24]. Note however that if H^{2}V reports *non-robust*, meaning the model is fragile in the specified neighbourhood, the conclusion is definitive. Note also that if in **Step 1** an overestimation of the Hölder constant is used, then H^{2}V is theoretically guaranteed to return sound results as the corollary below formalises.

Corollary 1. *Let H^* be the true Hölder constant. If for every iteration of* **Step 1**, *$\hat{H}_i \geq H^*$, then* H^{2}V *outputs* robust *iff* $\forall \boldsymbol{\theta} \in [0,1]^N : f(\boldsymbol{\theta}) > 0$.

Proof. The corollary is a direct consequence of Theorems 3.6 and 3.8 in [9]. □

Tight bounds for the Hölder constant are in general intractable to compute [9], while fast and loose bounds lead to major performance degradation of the verification procedure, as empirically analysed in the Appendix. In the light of this, H^{2}V relies on estimations of the constant as detailed in **Step 1**, but implements the following operational enhancements that remedy the potential underestimation of the constant.

Practical enhancements. To ensure high reliability, following convergence (*i.e.*, when the optimisation budget is reached), $\mathrm{H}^2\mathrm{V}$ iteratively increases the value of the global Hölder constant h_k and the number of neighbourhood intervals n_κ used in the estimation of the Hölder constant 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 constant will eventually trigger an escape from said minimum. To further enable high practical efficacy, $\mathrm{H}^2\mathrm{V}$ implements two global optimisation strategies [9]. 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 sub-intervals 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 the definition of the adaptive estimation of the Hölder constant $\hat{H}_j$ in **Step 1**). Taken together, these enhancements contribute towards achieving high efficiency and a very high degree of correctly answered verification problems.

4 Experimental Evaluation

Experimental Setup. Our experiments were conducted on a workstation equipped with a 16-core AMD Ryzen 9 9950X CPU, 192 GB of RAM, running Linux kernel 6.14.0-29-generic, and an NVIDIA RTX 5090 GPU with 32 GB of graphics memory. The implementation is in Python; the Hilbert space-filling curve mapping is implemented by using the hilbertcurve library [25]. The 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 reliability in practice. As we discuss below, our findings suggest that: the method scales to models such as vision transformers and video models that to our knowledge could not be previously verified and the implementation achieves the highest level of correctly answered verification queries.

In terms of geometric perturbations, we denote the rotation operation as $\mathrm{R}(\gamma)$, where the angle varies within the range $\pm\gamma$, and the scaling operation as $\mathrm{S}(\lambda)$, where the scaling factor ranges between $1 \pm \lambda$. Let $\mathrm{T}(t)$ 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 in terms of its *robust accuracy*, *i.e.*, the percentage of samples reported robust in the geometric neighbourhood considered. 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 $\mathrm{H}^2\mathrm{V}$ on image classification for large NNs, we benchmarked the robust accuracy obtained by the tool on 9 models from timm[1] of different sizes, ranging from 19M (Gmlp) to 300M (Large $\mathrm{ViT}_{16\times16}$) tunable parameters. These include several ResNet models up to ResNet152, trained to a good level of accuracy. The dataset used is ImageNet, with input sizes of either $3 \times 224 \times 224$ or $3 \times 299 \times 299$, depending on the model configuration. The verification queries consisted of any combination of input transformation consisting of rotation, translation and isotropic scaling with parameters 20°, 10%, and 10%, respectively. We set the timeout budget for each verification query to 1200s (20 minutes) and report the average runtime of $\mathrm{H}^2\mathrm{V}$ in seconds. The runtimes are computed with respect to the robust and non-robust cases and do not include the timeouts. To the best of 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, 21] can resolve such queries. We provide further benchmarking for completeness in the Appendix. Table 1 reports the results obtained for 500 ImageNet samples. We observe that in most cases $\mathrm{H}^2\mathrm{V}$ significantly outperforms GeoRobust in terms of finding more counterexamples whilst exhibiting a smaller percentage of timeouts. GeoRobust is shown to have superior performance on non-ResNet models in terms of robust accuracy. However, further analysis of these results indicate that GeoRobust often incorrectly reported a model as robust. Indeed, $\mathrm{H}^2\mathrm{V}$ identified several counterexamples (*i.e.*, 8 for Inception V3, 9 for ResNet34, 2 for ResNet50, 5 for ResNet101, 12 for ResNet152, 9 for Mixer,15 for Gmlp, 21 for Swin, 30 for ViT) to verification

[1] https://huggingface.co/docs/timm

Table 1: Evaluation results on 500 images from ImageNet against the perturbation combination (4 dimensions) of rotation (20°), translation (10%) and isotropic scaling (10%). The baseline performances are adopted from [22], except for ResNet-34, for which we rerun the experiment to obtain updated results due to changes in the timm (PyTorch Image Models) library.

Model	Input Size	Clean Acc (%)	No. Params (M)	Timeouts (%)		Robust Accuracy (%)		Average Runtime (s)
				GeoRobust*	H^2V	GeoRobust	H^2V	H^2V
Inception V3	$3 \times 299 \times 299$	73.4	24	4.0	0.6	24.2	23.0	150.51
ResNet34	$3 \times 299 \times 299$	72.0	22	4.2	1.0	27.8	26.0	101.55
ResNet50	$3 \times 299 \times 299$	78.4	26	22.9	1.8	31.1	40.8	253.97
ResNet101	$3 \times 299 \times 299$	80.0	45	6.0	2.2	48.2	47.0	430.27
ResNet152	$3 \times 299 \times 299$	79.6	60	7.2	2.0	46.2	46.8	477.01
Mixer	$3 \times 224 \times 224$	72.2	60	3.8	4.0	23.4	20.2	206.01
Gmlp	$3 \times 224 \times 224$	78.0	19	4.0	6.2	36.8	30.6	327.11
Swin	$3 \times 224 \times 224$	80.2	88	21.4	9.6	13.2	8.2	199.29
Large $\mathrm{ViT}_{16\times16}$	$3 \times 224 \times 224$	83.4	300	9.0	10.6	40.2	29.0	496.16

* GeoRobust's termination criterion is the query count; therefore, we here present the percentage of cases that remain undecided.

queries that were reported *robust* by GeoRobust. We suspect this is because GeoRobust's underlying global optimisation procedure can often use underestimations of the Lipschitz constant.

The results suggest that the ViT model considered is less robust than some ResNet models. This raises the question as to 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 H^2V, we now report the experimental results obtained when assessing 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 5 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], and C2D-R50 [33].

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

Model	Frame Length	Frame Rate	No. Params (M)	Clean Acc (%)	Timeouts (%)	Robust Acc (%)	Average Runtime (s)
X3D_M	16	5	3.79	73.0	3.0	37.0	1210.70
C2D-R50	8	8	24.33	73.0	3.0	36.0	768.88
I3D-R50	8	8	28.04	74.0	1.0	45.0	1118.28
R(2+1)D-R50	16	5	28.11	76.0	2.0	45.0	1874.70
Slow-R50	8	8	32.45	78.0	2.0	47.0	1238.82

For the evaluation, we randomly selected 100 videos from the dataset and evaluated the robustness of the models against perturbations applied to entire video (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. We set the timeout budget for each verification query to 3600s (60 minutes) and report the average runtime of H^2V in seconds. The runtimes are computed with respect to the robust and non-robust cases and do not include the timeouts. In terms of baselines, to the best of our knowledge, the only two verification methods applicable to video tasks are [34] and [35]. However, the former analyses the robustness of the extracted optical flow, rather than perturbing the RGB frames directly, thereby limiting its real-world applicability and comparability with our task. The latter can only scale to small NNs with small input sizes, hence it is not comparable with H^2V.

The results are reported in Table 2. It can be observed that H^2V was able to resolve a large proportion of the verification queries with a minimum rate of timeouts. To our knowledge, this is the first time that large video classifiers are evaluated for local robustness. In our results Slow-R50 and R(2+1)D-

Table 3: Soundness validation on SoundnessBench.

Benchmark	Input Dimensionality	No. Params	Tool	No. Robust	No. Non-Robust	No. Unknown	No. Unsound	Average Runtime (s)
CNN1	25-75	353K	$\alpha\beta$-CROWN	19	0	57	0	6.74
			PyRAT	12	0	64	0	3.50
			H^2V	27	0	49	0	66.06
CNN2	25-75	354K	$\alpha\beta$-CROWN	12	0	62	0	5.43
			PyRAT	5	0	69	0	15.32
			H^2V	16	0	58	0	63.31
CNN3	25-75	606K	$\alpha\beta$-CROWN	8	1	67	0	4.35
			PyRAT	5	0	71	0	9.63
			H^2V	12	0	64	0	62.87
CNN AvgPool	25-75	353K	$\alpha\beta$-CROWN	10	8	32	0	2.12
			PyRAT	6	0	54	0	18.80
			H^2V	12	0	48	0	65.47
CNN Tanh	25-75	353K	$\alpha\beta$-CROWN	0	1	37	0	0.71
			PyRAT	0	0	38	0	0.00
			H^2V	19	0	19	0	63.34
CNN Sigmoid	25-75	353K	$\alpha\beta$-CROWN	2	0	29	0	0.50
			PyRAT	1	0	30	0	0.36
			H^2V	19	0	12	0	63.80
MLP	10	3.13M	$\alpha\beta$-CROWN	20	3	48	0	0.79
			PyRAT	20	0	51	0	7.23
			H^2V	30	2	39	0	54.32

Table 4: Evaluation on TLL Verify Bench, a VNN-COMP benchmark.

Benchmark	Input Dimensionality	No. Params	Tool	No. Robust	No. Non-Robust	No. Unknown	No. Unsound	Average Runtime (s)
TLL Verify Bench	2	17k-67M	$\alpha\beta$-CROWN	15	17	0	0	37.93
			PyRAT	11	17	4	0	44.29
			H^2V	15	17	0	0	27.86

R50 achieved the highest robust accuracy (47.0% and 45.0%, respectively). These findings indicate that architectural refinements and expanding model capacity could potentially benefit robustness against geometric transformations.

Soundness Validation. We discussed in Sections 1 and 3 that since H^2V is based on global optimisation, it may return unsound results. This theoretical possibility can be mitigated, as it is routinely done in optimisation, by carefully choosing the optimisation parameters. The ablation studies of several relevant hyper-parameters, including the reliability parameter, the resolution parameter of the Hilbert curve, and the optimisation budget, are provided in the Appendix.

In what follows we evaluate the empirical soundness of H^2V. We do this in two ways. Firstly, we evaluate the results obtained by H^2V on SoundnessBench [8]. This is a recently released neural network verification benchmark, specifically designed for the validation of the correctness of verifiers by including the ground truth of the verification queries. Secondly, we report the results obtained by H^2V on low-dimensionality perturbations from VNN-COMP [4]. In this case the ground truth is not known, and thus, similarly to VNN-COMP, we compare the results against those produced by the SoA tools. In total, we evaluated the soundness of H^2V on 460 robustness queries. We found that all the results produced by H^2V on SoundnessBench were correct (*i.e.*, matching the ground truth provided), and all the results produced by H^2V on the VNN-COMP benchmarks were in line with those reported by $\alpha\beta$-CROWN [36].

We report the results obtained on SoundnessBench in Table 3, using a timeout of 100 seconds. 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. It has recently reported that several mainstream NN verifiers, including $\alpha\beta$-CROWN [36], NeuralSAT [37], and Marabou [38], answer some of the instances incorrectly [8]. In contrast, H^2V returned the correct result for all the queries, and achieved the highest number of verified queries. We refer to the Appendix for an exposition of the detailed performance of each method.

Lastly, Table 4 reports the results obtained on the TLL Verify Bench benchmark from VNN-COMP [4], using a default timeout of 600 seconds. The benchmark was selected because of the low-input dimensionality ($N = 2$) of the perturbations it includes, which makes it amenable to analysis via $\mathrm{H}^2\mathrm{V}$. We observe that $\mathrm{H}^2\mathrm{V}$ achieves the same verification results as $\alpha\beta$-CROWN (batch size 1) and PyRAT [39], while being more efficient and exhibiting no unsound cases.

5 Related work

An extensive body of literature exists on the verification of NN robustness against ℓ_p-bounded and other local perturbations; we refer to [2, 40] for surveys on the area. A variety of methods are used ranging from Mixed-Integer Linear Programming [41–44], to SMT [37, 38], abstract interpretation [39, 45–49], and branch-and-bound with symbolic interval propagation [36, 50–56]. All of these differ from $\mathrm{H}^2\mathrm{V}$ in that they are theoretically sound. This guarantee does not always translate into sound implementations since unavoidable floating point approximations may impact the correctness of the bounds generated by symbolic interval propagation methods [7, 8]. In contrast, outside the soundness envelope discussed in Section 3, $\mathrm{H}^2\mathrm{V}$ may in principle return "robust" for a model that admits attacks. We noted that this eventually is remote and can be mitigated by an appropriate choice of optimisation parameters as our experiments demonstrate.

In terms of scalability, the approaches cited above, notably symbolic interval propagation, outperform $\mathrm{H}^2\mathrm{V}$ for large dimensionality problems. However, as shown in the previous section, $\mathrm{H}^2\mathrm{V}$ considerably outperforms all SoA on geometric perturbations, including the approaches targeting geometric robustness directly [21, 22, 57–59] (see also a discussion in the Appendix).

Much closer to $\mathrm{H}^2\mathrm{V}$ are existing methods based on global optimisation [18, 24, 60]. The key difference between $\mathrm{H}^2\mathrm{V}$ 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 optimisation methods provably converge only if particular parameters can be chosen. However, this choice is closely related to establishing an upper bound of the Lipschitz constant. This is normally intractable for large models and only estimations can be used in practice, thereby potentially resulting in unsound results, as we empirically observed.

6 Conclusions

We presented $\mathrm{H}^2\mathrm{V}$, a novel method for the robustness validation of NNs against geometric perturbations. We demonstrated that $\mathrm{H}^2\mathrm{V}$ outperforms SoA robustness verification methods against geometric perturbations and scales to vision classification models of hundreds of millions of tunable parameters and large inputs, enabling, for the first time to our knowledge, the validation of large video classifiers. These results enable the rigorous validation of present vision systems including transformer-based architectures. We noted that, differently from several present verification methods, the theoretical soundness of $\mathrm{H}^2\mathrm{V}$ cannot be guaranteed in all cases, but we presented the reasons why we do not regard this as a limitation in practice.

Acknowledgements

Alessio Lomuscio acknowledges partial support from the Royal Academy of Engineering via a Chair of Emerging Technologies. All authors are grateful to the anonymous referees for valuable comments that led to considerable improvements to this article.

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):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 *Proceedings of the 29th International Conference on Computer Aided Verification (CAV17)*, Lecture notes in computer science, 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 2412.19985*, 2024.

[5] 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.

[6] Alex Serban, Erik Poll, and Joost Visser. Adversarial examples on object recognition. *ACM Computing Surveys*, 53(3):1–38, 2021.

[7] Kai Jia and Martin Rinard. Exploiting verified neural networks via floating point numerical error. *arXiv preprint arXiv:2003.03021*, 2020.

[8] Xingjian Zhou, Keyi Shen, Andy Xu, Hongji Xu, Cho-Jui Hsieh, Huan Zhang, and Zhouxing Shi. SoundnessBench: A soundness benchmark for neural network verifiers. *arXiv preprint arXiv:2412.03154*, 2025.

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

[10] Daniela Lera and Yaroslav 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, 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*. Springer, 2014.

[14] Giuseppe Peano. Sur une courde qui remplit touteune aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[15] David Hilbert. Ueber die stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, 38(3):459–460, 1891.

[16] Roman G Strongin and Yaroslav D Sergeyev. *Global optimization with non-convex constraints*. Springer, 2013.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 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 (IJCAI18)*, page 2651–2659. ijcai.org, 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, and Koray Kavukcuoglu. Spatial transformer networks. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NeurIPS15)*, pages 2017–2025. Curran Associates Inc., 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 *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI24)*, 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 37th AAAI Conference on Artificial Intelligence (AAAI23)*, pages 15197–15205. AAAI Press, 2023.

[23] Eric Gourdin, Brigitte Jaumard, and Rachid Ellaia. Global optimization of hölder functions. *Journal of Global Optimization*, 8(4):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), 2025.

[25] Gabriel Altay. `hilbertcurve` (python package). `https://pypi.org/project/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 *Proceedings of the 39th International Conference on Machine Learning (ICML22)*, 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, Mustafa Suleyman, and Andrew Zisserman. 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 (ACMMM21)*, 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 2017 IEEE International Conference on Computer Vision (ICCV17) 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 (CVPR18)*, 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 (CVPR20)*, 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 (CVPR17)*, pages 6299–6308, 2017.

[33] 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 (CVPR18)*, pages 7794–7803, 2018.

[34] 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 (CVPR20)*, pages 311–320, 2020.

[35] Samuel Sasaki, Diego Manzanas Lopez, Preston K Robinette, and Taylor T Johnson. Robustness verification of video classification neural networks. In *Proceedings of the 13th IEEE/ACM International Conference on Formal Methods in Software Engineering (FormaliSE25)*, pages 22–33. IEEE, 2025.

[36] Huan Zhang, Shiqi Wang, Kaidi Xu, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS22)*, pages 1656–1670. Curran Associates Inc., 2022.

[37] Hai Duong, Thanhvu Nguyen, and Matthew Dwyer. A DPLL(T) framework for verifying deep neural networks. *arXiv preprint arXiv:2307.10266*, 2023.

[38] 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 *Proceedings of the 36th International Conference on Computer Aided Verification (CAV24)*, pages 249–264. Springer, 2024.

[39] Augustin Lemesle, Julien Lehmann, and Tristan Le Gall. Neural network verification with PyRAT. *arXiv preprint arXiv:2410.23903*, 2025.

[40] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, and Mykel J Kochenderfer. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*, 2019.

[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 1706.07351*, 2017.

[43] 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 34th AAAI Conference on Artificial Intelligence (AAAI20)*, pages 3291–3299. AAAI Press, 2020.

[44] Panagiotis Kouvaros and Alessio Lomuscio. Towards scalable complete verification of relu neural networks via dependency-based branching. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2643–2650. ijcai.org, 2021.

[45] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. AI^2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy (SP18)*, pages 3–18. IEEE, 2018.

[46] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS18)*, page 10825–10836. Curran Associates Inc., 2018.

[47] 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.

[48] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open-and closed-loop neural network verification using polynomial zonotopes. In *Proceedings of the 15th NASA Formal Methods Symposium (NFM23)*, pages 16–36. Springer, 2023.

[49] Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T Johnson. Nnv 2.0: The neural network verification tool. In *Proceedings of the 35th International Conference on Computer Aided Verification (CAV23)*, pages 397–412. Springer, 2023.

[50] 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 *Proceedings of the 9th International Conference on Learning Representations (ICLR21)*. OpenReview.net, 2021.

[51] 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.

[52] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS18)*, page 4944–4953. Curran Associates Inc., 2018.

[53] 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. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NeurIPS21)*, pages 29909–29921. Curran Associates Inc., 2021.

[54] Patrick Henriksen and Alessio Lomuscio. DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2549–2555. ijcai.org, 2021.

[55] Panagiotis Kouvaros, Benedikt Brückner, Patrick Henriksen, and Alessio Lomuscio. Dynamic back-substitution in bound-propagation-based neural network verification. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI25)*, pages 27383–27391. AAAI Press, 2025.

[56] Benedikt Brückner and Alessio Lomuscio. Verification of neural networks against convolutional perturbations via parameterised kernels. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI25)*, pages 27215–27223. AAAI Press, 2025.

[57] 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 (CCS21)*, pages 535–557. ACM, 2021.

[58] Zhongkai Hao, Chengyang Ying, Yinpeng Dong, Hang Su, Jian Song, and Jun Zhu. GSmooth: Certified robustness against semantic transformations via generalized randomized smoothing. In *Proceedings of the 39th International Conference on Machine Learning (ICML22)*, pages 8465–8483. PMLR, 2022.

[59] Mislav Balunovic, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin Vechev. Certifying geometric robustness of neural networks. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS19)*, pages 15313 – 1532. Curran Associates Inc., 2019.

[60] Chi Zhang, Wenjie Ruan, and Peipei Xu. Reachability analysis of neural network control systems. *arXiv preprint arXiv:2301.12100*, 2023.

[61] Rem Yang, Jacob Laurel, Sasa Misailovic, and Gagandeep Singh. Provable defense against geometric transformations. In *Proceedings of the 11th International Conference on Learning Representations (ICLR23)*. OpenReview.net, 2023.

[62] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems (NeurIPS20)*, pages 1129–1141. Curran Associates Inc., 2020.

[63] Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh. Would mega-scale datasets further enhance spatiotemporal 3D CNNs? *arXiv preprint arXiv:2004.04968*, 2020.

[64] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[65] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[66] Donald R Jones and Joaquim R R A Martins. The DIRECT algorithm: 25 years later. *Journal of Global Optimization*, 79(3):521–566, 2021.

A Proof of Theorem 1

Lemma 1. *([10, Theorem 2.1]) Let l_h^* be the solution of the lower bound along the Hilbert curve $h_{N,m}(x)$ for $f(\boldsymbol{\theta})$ satisfying the Lipschitz condition with constant L:*

$$l_h^\star \leq \min f(h_{N,m}(x)) = \min \tilde{f}(x), \quad x \in [0,1]. \tag{11}$$

Considering the infinite optimisation budget, i.e.*, $\epsilon = 0$, then the true lower bound for the multi-dimensional problem* $\min f(\boldsymbol{\theta})$ *in the original space over the entire region* $[0,1]^N$*:*

$$l_h^\star - L \cdot 2^{-(m+1)}\sqrt{N} \leq l^*. \tag{12}$$

Proof can be found in [10, Thm. 2.1]. □

Theorem 1. *Let l_h^* 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*

$$l_h^\star - L \cdot 2^{-(m+1)}\sqrt{N} - H \cdot (\epsilon/2)^{1/N} \leq l^*, \tag{13}$$

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 first calibration term directly comes from Lemma. 1. Let $\tilde{f} : [0,1] \to \mathbb{R}$ satisfy the Hölder condition with exponent $\alpha = 1/N$ and constant $H > 0$:

$$|\tilde{f}(x) - \tilde{f}(y)| \leq H\,|x-y|^{1/N}, \quad \forall x, y \in [0,1].$$

Assume that during H^{2}V's optimisation process, Theorem 2 in the main manuscript has been satisfied, *i.e.*, $r > r^\star$ such that $\hat{H}_i \geq H_i$ for each interval, the algorithm terminates when the selected interval $[x_{i-1}, x_i]$ has length $\delta_i = |x_i - x_{i-1}| \leq \epsilon$. Let

$$x^* = \arg\min_{x \in [0,1]} \tilde{f}(x), \quad \hat{x} = \arg\min\{\tilde{f}(x_i) : \text{sampled endpoints } x_i\},$$

and denote $\tilde{f}^* = \tilde{f}(x^*)$, $\hat{\tilde{f}} = \tilde{f}(\hat{x})$. Let ξ represent the endpoint of the interval $[x_{i-1}, x_i]$ that is closest to the true minimizer x^*. Then, upon termination [9], the true minimizer x^* will only fall into the chosen interval. By definition of ξ as the nearer endpoint, we have

$$|x^* - \xi| = \min\{|x^* - x_{i-1}|, |x^* - x_i|\} \leq \frac{|x^* - x_{i-1}| + |x^* - x_i|}{2} = \frac{\delta_i}{2} \leq \frac{\epsilon}{2}. \tag{14}$$

From the Hölder condition for the function $\tilde{f}$:

$$|\tilde{f}(\xi) - \tilde{f}(x^*)| \leq H\,|\xi - x^*|^{1/N} \leq H\Big(\frac{\epsilon}{2}\Big)^{1/N}. \tag{15}$$

Since $\hat{\tilde{f}} = \min_k \tilde{f}(x_k) \leq \tilde{f}(\xi)$,

$$\hat{\tilde{f}} - \tilde{f}^* \leq |\tilde{f}(\xi) - \tilde{f}(x^*)| \leq H\Big(\frac{\epsilon}{2}\Big)^{1/N}. \tag{16}$$

Therefore, $H \cdot (\epsilon/2)^{1/N}$ is incorporated into the lower bound in Eq. (13) as the second calibration term, thereby completing the proof. □

Summary: The first 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 H^{2}V to high-dimensional input domains.

B Ablation Study

In this section we report several ablation studies for some hyper-parameters used in our method. Specifically, we conducted ablation studies on the reliability parameter r on two benchmarks with ground-truth: TLL Verify Bench from VNN-COMP (2 dimensions) and a MLP5 benchmark from SoundnessBench (10 dimensions). The timeout is set to 100 seconds. We report the counts of Robust, Non-Robust, and Unknown cases, respectively, along with the average runtime in seconds. The runtimes are computed with respect to the robust and non-robust cases and do not include the timeouts. The results show that $\mathrm{H}^2\mathrm{V}$ outputs no unsound results for any value of $r > 1$. However, we note an increase difficulty to answer verification queries with bigger values of r. Empirical results from the optimisation literature [11] also suggest that the use of a small value close to 1, *e.g.*, 1.3, yields close to optimal solutions. This is aligned with the observations in [11].

In addition, the computational complexity of modern libraries providing the Hilbert space mapping grows linearly in m, so any value of $m > 30$, which already provides small calibration errors (see mathematical expression (Eq. 10) in the paper), would be adequate to use. We report an ablation study on m in Table 6 which confirms this intuition: any value of $m > 10$ resolved all verification queries with stable overheads.

Also note that the optimisation budget corresponds to the typical optimisation gap present in all convergent algorithms. As shown in Table 7, a smaller value generally enables the method to solve more verification queries, as it allows exploration of a finer-grained search space. Its value is thus contingent to available computation and temporal resources.

Table 5: Ablation study on the reliability parameter r.

Benchmark	Perturbation Dimensionality	No. Params	r	No. Robust (Sound)	No. Non-Robust	No. Unknown	Average Runtime (s)
TLL Verify Bench	2	17k-67M	1.1	15 (15)	17	0	26.45
			1.3	15 (15)	17	0	27.81
			1.5	15 (15)	17	0	26.06
			3	12 (12)	17	3	24.51
			6	6 (6)	17	9	17.60
			10	3 (3)	17	12	10.87
SoundnessBench MLP5 (epsilon 0.2)	10	3.13M	1.1	10 (10)	2	2	47.08
			1.3	10 (10)	2	2	50.81
			1.5	9 (9)	2	3	39.27
			3	0 (0)	2	12	4.04
			6	0 (0)	1	13	68.20
			10	0 (0)	0	14	-

Table 6: Ablation study on the Hilbert curve resolution parameter m.

Benchmark	Perturbation Dimensionality	No. Params	m	No. Robust (Sound)	No. Non-Robust	No. Unknown	Average Runtime (s)
TLL Verify Bench	2	17k-67M	10	0 (0)	17	15	1.65
			30	15 (15)	17	0	25.22
			50	15 (15)	17	0	26.06
			70	15 (15)	17	0	26.57
			90	15 (15)	17	0	27.39
SoundnessBench MLP5 (epsilon 0.2)	10	3.13M	10	10 (10)	1	3	44.85
			30	10 (10)	1	3	56.35
			50	10 (10)	2	2	50.81
			70	10 (10)	1	3	52.29
			90	9 (9)	2	3	47.92

C Additional Experimental Evaluation

We here provide additional experimental results for image and video tasks.

Table 7: Ablation study on the optimisation budget parameter ϵ.

Benchmark	Perturbation Dimensionality	No. Params	ϵ	No. Robust (sound)	No. Non-Robust	No. Unknown	Average Runtime (s)
TLL Verify Bench	2	17k-67M	$1e-3$	0 (0)	17	15	1.60
			$1e-6$	15 (15)	17	0	25.91
			$1e-9$	15 (15)	17	0	26.60
			$1e-13$	15 (15)	17	0	26.22
			$1e-15$	15 (15)	17	0	26.06
SoundnessBench MLP5 (epsilon 0.2)	10	3.13M	$1e-3$	0 (0)	0	14	-
			$1e-6$	5 (5)	0	9	57.16
			$1e-9$	10 (10)	2	2	45.10
			$1e-13$	10 (10)	2	2	46.23
			$1e-15$	10 (10)	2	2	50.81

C.1 Verification on Image Classification

We return to the simple case where we consider geometric perturbation in one dimension only. Table 8 reports the results obtained on the MNIST and CIFAR-10 datasets. H^2V outperforms other methods, including TSS [57], GeoRobust [22], DeepG [59], and GSmooth [58].

Table 8: Comparing with baseline methods on MNIST/CIFAR-10 against rotation and scaling. Baselines performance is adopted from [22, 57, 58].

Dataset	Model	Clean Acc	Pert	Adversarial Accuracy (%)				Robust Accuracy (%)					Average Runtime (s)
				TSS	GeoRobust	GridSearch	H^2V	DeepG	GSmooth	TSS	GeoRobust	H^2V	H^2V
MINST	Small CNN	99.4	R(50°)	98.2	98.2	98.2	98.2	85.8	95.7	97.4	98.2	98.2	55.38
		99.4	S(30%)	99.2	99.2	99.2	99.2	85.0	95.9	97.2	99.2	99.2	83.45
CIFAR-10	ResNet110	84.0	R(10°)	76.4	74.8	74.8	74.8	62.5	65.6	70.6	74.6	74.8	216.35
		81.2	R(30°)	69.4	66.6	66.4	66.4	10.6	-	63.6	66.2	66.4	169.37
		80.8	S(30%)	67.0	63.4	63.4	63.4	00.0	54.3	58.8	62.8	63.4	170.23

Additionally, we conducted experiments on Tiny ImageNet to evaluate the WideResNet (with 7.94M parameters), adopted from Certified Geometric Training (CGT) [61], which uses auto_LiRPA [62] to perform geometric robustness verification. Experimental results in Table 9 illustrate that H^2V verifies all images and returns any counterexample found.

Table 9: Comparing with baseline method CGT on the Tiny-ImageNet against rotation and scaling perturbations.

Perturbation	Clean Accuracy (%)	Timeouts (%)		Robust Accuracy (%)		Average Runtime (s)
		CGT	H^2V	CGT	H^2V	H^2V
S(2%)	33.10	4.06	0.00%	21.44	25.50	48.99
R(5°)	32.01	3.98	0.00%	17.49	21.47	45.58

C.2 Verification on Video Classification

We reported the results on several large 3D-NNs on the Kinetics dataset in the main manuscript. Here we report results on geometric robustness validation on the UCF101 dataset.

Long-length video classification often requires segmentation into shorter clips, achieved by considering $\boldsymbol{x} = n \times [v_{\text{clip}}]$. Two metrics are often used for assessing model performance: clip-level accuracy and video-level accuracy. Clip-level accuracy records the proportion of correctly predicted clips across all videos. Video-level accuracy is computed by aggregating the predicted probabilities of individual clips for each video and using the accumulated probabilities to compute the final prediction accuracy at the video level. Let $V(\boldsymbol{x}, \boldsymbol{\theta})$ represent the perturbed video/clip, we then analyse the video classifier's robustness by simulating the effects of geometric distortions, including tilted camera orientation (rotation), positional shifts (translation), and zoom variations (scaling). We focus on geometric robustness against frame-agnostic perturbations at the video level. In this setting, the same transformation $\boldsymbol{\theta}$ is applied consistently across all frames, and we evaluate the classifier's ability to handle such uniform geometric distortions. This setup captures the overall impact of static transformations, such as those caused by incorrect camera positioning. The experiments conducted on the Kinetics dataset presented in the main manuscript also belong to this category.

Analogously, the local geometric robustness problems for video classification can be formulated as checking whether $\forall \boldsymbol{\theta} \in \boldsymbol{\Theta} : f(\boldsymbol{\theta}) > 0$. In our setting this is solved by evaluating $\forall x \in [0, 1] : \tilde{f}(x) > 0$ in the corresponding optimisation problem. We here focus on a 3D-ResNet50 model, adapted from [63], to explore its robustness on the UCF101 dataset [64]. The model processes entire videos using a frame length of 16 and a frame rate of 1, with 46.4M network parameters. Since the frame-agnostic perturbations are applied uniformly across all frames in a video, the number of perturbation dimension ranges from 1 to 4.

We select the first two videos of each class in the test set, which consists of 202 videos in total. Since we are not aware of any previous work supporting this setup, to provide a baseline comparison, we adapted another global optimisation based verification method, DeepGO [18] with a pre-defined Lipschitz constant $L = 8$ (follow their settings), PGD [65], and as two strong baselines for comparison. We controlled the termination of all methods by setting a maximum number of queries $K = 100,000$ and timeout as 2400 seconds (40 minutes). The 3D-ResNet50 model we are adopting achieves 87.13% clean accuracy across the 202 videos. Detailed results against various consistent geometric perturbations at video level are summarised in Table 10. As we can see, H^{2}V obtains the best adversarial accuracy and robust accuracy and these values coincide. This means that the method could solve all queries for this model and inputs with no unknowns. We are also able to establish that for the parameters analysed, rotation has the highest impact on the fragility of the model. As we discuss in the main part of the paper, global optimisation-based verification methods without guarantees on the soundness of the upper bound of the Lipschitz constant may converge to a local minimum. As a consequence, they may potentially derive unsound results, as indicated in Table 10.

Table 10: Evaluation of static geometric robustness for a 3D-ResNet50 model with 87.13% clean video accuracy on UCF101 dataset. Below R, S, T refer to R($20°$), S(10%), T(10%), respectively.

Perturbation	Perturbation Dimensionality	Adversarial Accuracy (%)		Robust Accuracy (%)		Average Runtime (s)
		PGD	H^{2}V	DeepGO	H^{2}V	H^{2}V
R	1	51.98	43.07	45.05	43.07	458.10
T	2	76.73	66.83	70.30	66.83	634.15
S	1	75.74	74.75	75.74	74.75	929.69
R + T	3	48.02	34.65	37.13	34.65	358.03
R + S	2	50.10	35.64	36.14	35.64	342.10
T + S	3	72.77	58.91	61.39	58.91	613.48
R + T + S	4	40.10	22.77	25.25	22.77	286.33

Figure 2 illustrates the impact of rotation (R), translation (T), scaling (S), and their combination on the static geometric robustness in video classification under different perturbation magnitudes. Note that H^{2}V is able to solve all the robustness verification among the following perturbation settings. A clear trend emerges, indicating that as the perturbation magnitude increases, robust accuracy consistently declines. While individual transformations already contribute to performance degradation, their combinations exacerbate the effect, leading to even more significant accuracy drops. Among these three geometric factors, scaling (S) appears to be the most stable than shifting and angular transformations, as its impact on accuracy degradation is relatively lower compared to rotation and translation.

C.3 Detailed Experimental Results on SoundnessBench and VNN-COMP

Table 11 reports detailed verification results on SoundnessBench [8] for $\alpha\beta$-CROWN[2], PyRAT[3], and H^{2}V, respectively, with default timeout setting 100 seconds for each instance. It can be seen that,

[2]https://github.com/Verified-Intelligence/alpha-beta-CROWN

[3]https://git.frama-c.com/pub/pyrat

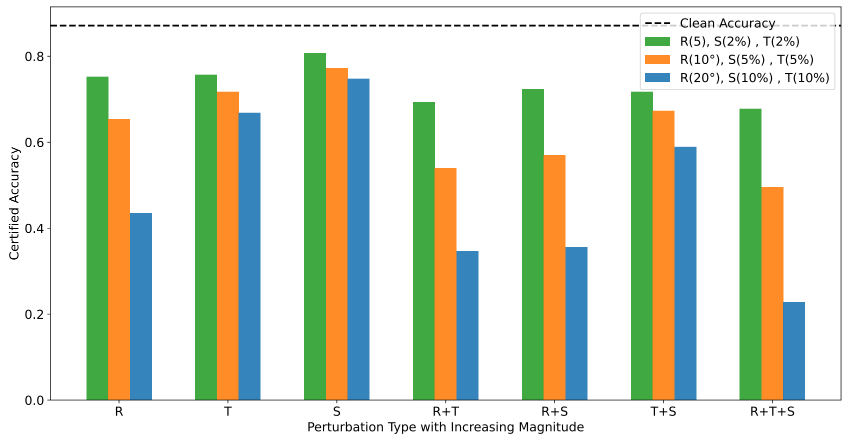

Figure 2: Robust accuracy decreases as perturbation magnitude increases.

differently from previously reported, in our tests, all three tested tools now pass the soundness tests. Previously [8] identified unsound results in $\alpha\beta$-CROWN [36]. It can be observed that $\mathrm{H}^2\mathrm{V}$ provides more positive answers for the verification queries, and finds counterexamples for queries that cannot be solved by other tools. This highlights its effectiveness in NN verification.

For additional benchmarking from VNN-COMP [4], we consider TLL Verify Bench, which has the low-input dimensionality of 2 of the perturbation. The TLL Verify Bench consists of Two-Level Lattice (TLL) NNs; it has been proposed as a means of comparing TLL-specific verification algorithms with general-purpose NN verification algorithms. We used the default timeout setting, 600 seconds, described in VNN-COMP 2023 [4]. $\mathrm{H}^2\mathrm{V}$ achieved the same SOA performance on these two benchmarks as $\alpha\beta$-CROWN.

Table 11: Soundness validation on SoundnessBench. Numbers in **bold** denote that counterexamples were found.

<table>
<tr><th colspan="3">Perturbation Settings</th><th rowspan="2">No. Sample
(Clean | Unveri)</th><th colspan="3">Results for Clean Instances
(No. UNSAT | Unknown)</th><th colspan="3">Results for Unverifiable Instances
(No. UNSAT | SAT | Unknown)</th><th colspan="3">Average Runtime (s)</th></tr>
<tr><th>Model Name</th><th>ϵ</th><th>Input Size</th><th>$\alpha\beta$-CROWN</th><th>PyRAT</th><th>$\mathrm{H}^2\mathrm{V}$</th><th>$\alpha\beta$-CROWN</th><th>PyRAT</th><th>$\mathrm{H}^2\mathrm{V}$</th><th>$\alpha\beta$-CROWN</th><th>PyRAT</th><th>$\mathrm{H}^2\mathrm{V}$</th></tr>
<tr><td>CNN 1 Conv</td><td>0.2</td><td>1×5×5</td><td>10 | 9</td><td>10 | 0</td><td>10 | 0</td><td>10 | 0</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0.65</td><td>1.02</td><td>59.92</td></tr>
<tr><td>CNN 1 Conv</td><td>0.2</td><td>3×5×5</td><td>10 | 9</td><td>9 | 1</td><td>0 | 10</td><td>10 | 0</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>13.51</td><td>15.87</td><td>73.68</td></tr>
<tr><td>CNN 1 Conv</td><td>0.5</td><td>1×5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>7 | 3</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>-</td><td>-</td><td>63.97</td></tr>
<tr><td>CNN 1 Conv</td><td>0.5</td><td>3×5×5</td><td>10 | 8</td><td>0 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 0 | 8</td><td>0 | 0 | 8</td><td>0 | 0 | 8</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>CNN 2 Conv</td><td>0.2</td><td>1×5×5</td><td>10 | 7</td><td>9 | 1</td><td>5 | 5</td><td>10 | 0</td><td>0 | 0 | 7</td><td>0| 0 | 7</td><td>0 | 0 | 7</td><td>2.11</td><td>15.32</td><td>58.74</td></tr>
<tr><td>CNN 2 Conv</td><td>0.2</td><td>3×5×5</td><td>10 | 7</td><td>3 | 7</td><td>0 | 10</td><td>6 | 4</td><td>0 | 0 | 7</td><td>0 | 0 | 7</td><td>0 | 0 | 7</td><td>15.38</td><td>-</td><td>70.92</td></tr>
<tr><td>CNN 2 Conv</td><td>0.5</td><td>1×5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>CNN 2 Conv</td><td>0.5</td><td>3×5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>CNN 3 Conv</td><td>0.2</td><td>1×5×5</td><td>10 | 10</td><td>8 | 2</td><td>5 | 5</td><td>9 | 1</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>4.79</td><td>9.63</td><td>64.14</td></tr>
<tr><td>CNN 3 Conv</td><td>0.2</td><td>3×5×5</td><td>10 | 7</td><td>0 | 10</td><td>0 | 10</td><td>1 | 9</td><td>0 | 0 | 7</td><td>0 | 0 | 7</td><td>0 | 0 | 7</td><td>-</td><td>-</td><td>69.45</td></tr>
<tr><td>CNN 3 Conv</td><td>0.5</td><td>1×5×5</td><td>10 | 9</td><td>0 | 10</td><td>0 | 10</td><td>2 | 8</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>-</td><td>-</td><td>53.89</td></tr>
<tr><td>CNN 3 Conv</td><td>0.5</td><td>3×5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 1 | 9</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0.82</td><td>-</td><td>-</td></tr>
<tr><td>CNN AvgPool</td><td>0.2</td><td>1×5×5</td><td>10 | 0</td><td>10 | 0</td><td>6 | 4</td><td>10 | 0</td><td>0 | 0 | 0</td><td>0 | 0 | 0</td><td>0 | 0 | 0</td><td>3.21</td><td>18.80</td><td>62.89</td></tr>
<tr><td>CNN AvgPool</td><td>0.2</td><td>3×5×5</td><td>10 | 1</td><td>0 | 10</td><td>0 | 10</td><td>1 | 9</td><td>0 | 0 | 0</td><td>0 | 0 | 1</td><td>0 | 0 | 1</td><td>-</td><td>-</td><td>86.15</td></tr>
<tr><td>CNN AvgPool</td><td>0.5</td><td>1×5×5</td><td>10 | 9</td><td>0 | 10</td><td>0 | 10</td><td>1 | 9</td><td>0 | 0 | 0</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>-</td><td>-</td><td>70.52</td></tr>
<tr><td>CNN AvgPool</td><td>0.5</td><td>3× 5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 10</td><td>0 | 8 | 2</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0.74</td><td>-</td><td>-</td></tr>
<tr><td>CNN Tanh</td><td>0.2</td><td>1×5×5</td><td>10 | 8</td><td>0 | 10</td><td>0 | 10</td><td>10 | 0</td><td>0 | 1 | 7</td><td>0 | 0 | 8</td><td>0 | 0 | 8</td><td>0.71</td><td>-</td><td>58.04</td></tr>
<tr><td>CNN Tanh</td><td>0.2</td><td>3×5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>9 | 1</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>-</td><td>-</td><td>69.23</td></tr>
<tr><td>CNN Sigmoid</td><td>0.2</td><td>1×5×5</td><td>10 | 1</td><td>2 | 8</td><td>1 | 9</td><td>9 | 1</td><td>0 | 0 | 1</td><td>0 | 0 | 1</td><td>0 | 0 | 1</td><td>0.50</td><td>0.36</td><td>56.45</td></tr>
<tr><td>CNN Sigmoid</td><td>0.2</td><td>3×5×5</td><td>10 | 10</td><td>0 | 10</td><td>0 | 10</td><td>10 | 0</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>0 | 0 | 10</td><td>-</td><td>-</td><td>70.43</td></tr>
<tr><td>MLP 4 Hidden</td><td>0.2</td><td>10</td><td>10 | 9</td><td>10 | 0</td><td>10 | 0</td><td>10 | 0</td><td>0 | 2 | 7</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0.70</td><td>2.10</td><td>49.46</td></tr>
<tr><td>MLP 4 Hidden</td><td>0.5</td><td>10</td><td>10 | 9</td><td>1 | 9</td><td>1 | 9</td><td>3 | 7</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>1.60</td><td>60.10</td><td>76.00</td></tr>
<tr><td>MLP 5 Hidden</td><td>0.2</td><td>10</td><td>10 | 4</td><td>8 | 2</td><td>8 | 2</td><td>10 | 0</td><td>0 | 1 | 3</td><td>0 | 0 | 4</td><td>0 | 2 | 2</td><td>0.78</td><td>5.84</td><td>52.16</td></tr>
<tr><td>MLP 5 Hidden</td><td>0.5</td><td>10</td><td>10 | 9</td><td>1 | 9</td><td>1 | 9</td><td>7 | 3</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>0 | 0 | 9</td><td>1.31</td><td>16.83</td><td>55.66</td></tr>
</table>

D Comparison with the State-of-the-Art Methods

In this section, we compare $\mathrm{H}^2\mathrm{V}$ with $\alpha\beta$-CROWN, the SoA tool from VNN-COMP [4]. The verification queries from our main experiments cannot be used for this comparison. Firstly, the tools do not support the `affine_grid` and `grid_sample` required for the geometric perturbations via the Spatial Transformer Networks. Secondly, the tools do not scale to the sizes of the models considered in the main experiments. To illustrate this, we used the `ResNet18_cifar` model from the $\alpha\beta$-CROWN repository, and considered different model sizes by adjusting the width of the layers through thje *in_planes* parameter of the benchmark. For the verification of the models, we constructed one-pixel perturbations (corresponding to three dimensions), serving as a comparable low-dimensional proxy with $\mathrm{H}^2\mathrm{V}$ capabilities. We then performed verification queries for $\epsilon = 0.1$ and $\epsilon = 0.3$.

Table 12 reports the results obtained on this simplified setting. We observe that $\alpha\beta$-CROWN verifies the smallest case at $\epsilon = 0.1$ but either timeouts or runs out of memory (OOM) as ϵ increases or the model grows. In particular, $\alpha\beta$-CROWN runs out of memory (OOM) on a relatively small model with 175,802 parameters. In contrast, $\mathrm{H}^2\mathrm{V}$ consistently returns robust with lower and more stable memory consumption. In conclusion, SoA methods cannot scale to models of tens/hundreds of millions of parameters as we do here, or to ImageNet-scale inputs with geometric perturbations.

Table 12: Verification results against a one-pixel perturbation for CIFAR10 models.

Model	In_planes	No. Params	Perturbation Magnitude	Perturbation Dimensionality	Tool	Result	Peak Memory (MB)	Runtime (s)
ResNet18_cifar	2	11,270	0.1	3	PGD	Unknown	656	12.75
					$\alpha\beta$-CROWN	Robust	644	0.73
					$\mathrm{H}^2\mathrm{V}$	Robust	636	68.33
	2	11,270	0.3	3	PGD	Unknown	656	13.32
					$\alpha\beta$-CROWN	Unknown	4298	Timeout
					$\mathrm{H}^2\mathrm{V}$	Robust	636	61.02
	8	175,802	0.1	3	PGD	Unknown	678	14.16
					$\alpha\beta$-CROWN	Unknown	OOM	Timeout
					$\mathrm{H}^2\mathrm{V}$	Robust	638	56.54

E Performance Analysis of $\mathrm{H}^2\mathrm{V}$ With Soundness Guarantees

In Section 3, we noted that if Step 1 of $\mathrm{H}^2\mathrm{V}$ uses an overestimation of the Hölder constant, then the method is theoretically guaranteed to produce sound results. Here, we use $\mathrm{H}^2\mathrm{V}^*$ to denote $\mathrm{H}^2\mathrm{V}$ with such overestimations. We then compare the scalability of $\mathrm{H}^2\mathrm{V}$ and $\mathrm{H}^2\mathrm{V}^*$, allowing us to empirically identify the practical limitations of $\mathrm{H}^2\mathrm{V}^*$.

For this, we use the CIFAR10 and RESNET models from VNNCOMP and 1-pixel perturbations. Table13 reports the results. These show that $\mathrm{H}^2\mathrm{V}^*$ is able to verify the models associated with small Lipschitz constants but timeouts when the constant exceeds a certain value. In contrast, $\mathrm{H}^2\mathrm{V}$ validates all models and scales to models with significantly larger Lipschitz constants (*e.g.*, around 8e+50 for ResNet34) and much larger models (over 300M parameters), as reported in the paper. We stress that, unlike much of the existing verification literature, model size itself is not the primary bottleneck in our setting. This is instead the value of the Lipschitz constant, as these experiments demonstrate.

F Further implementation details of $\mathrm{H}^2\mathrm{V}$

In the main manuscript, we have described the optimisation process of our method in detail. The pseudocode of $\mathrm{H}^2\mathrm{V}$ is also outlined in Algorithm 1. In the algorithm, we use K to control the maximum number of queries considered. The algorithm initiates the reliability parameter with $r = 1.3$, as recommended by [11]. When the algorithm converges (*i.e.*, when it reaches the optimisation budget), the size of the neighbourhood is increased $n_\kappa \leftarrow n_\kappa + 1$, and the value of the global Hölder constant iteratively loosened (*i.e.*, $h_k \leftarrow h_k \cdot 1.3$), until a different interval is selected in **Step 16** of the algorithm. This process is repeated until the the optimisation budget is reached for the same interval for 25 times, at which point the estimated lower bound is used to answer the geometric

Table 13: Verification results on different models on the CIFAR10 examples.

Model	Upper bound of Lipschitz Constant	No. Params	Perturbation Magnitude	Perturbation Dimensionality	Method	Result	Runtime (s)
marabou-cifar10/cifar10_small	1.2888e+02	2,456	0.1	3	H^2V	Robust	61.58
					H^2V^*	Robust	2.68
marabou-cifar10/cifar10_small	2.0159e+02	9,008	0.1	3	H^2V	Robust	53.37
					H^2V^*	Robust	4.49
marabou-cifar10/cifar10_small	4.2458e+02	34,400	0.1	3	H^2V	Robust	65.02
					H^2V^*	Robust	161.73
cifar10_resnet/resnet2b	2.2218e+08	112,006	0.1	3	H^2V	Robust	56.98
					H^2V^*	Unknown	Timeout
cifar10_resnet/resnet4b	7.8005e+13	123,734	0.1	3	H^2V	Robust	48.73
					H^2V^*	Unknown	Timeout

* H^2V^*: H^2V with an upper bound of Lipschitz constant, hence an upper bound of Hölder constant as well.

robustness query. As shown in the experimental results, this solution provide a good approximation for the unknown Hölder constant.

G Additional Related Work

Besides what previously referenced, further approaches have been put forward for the verification of local robustness against geometric perturbations. We here compare these with H^2V.

By constructing the set of allowed perturbation values using an ℓ_p bound for each pixel and replacing it with a convex relaxation, [19, 47] provide an over-approximation for the solution but result in loose performance bounds.

TSS [57] and GSmooth [58] demonstrate the potential of randomised smoothing to certify robustness against individual geometric transformations. However, these approaches provide statistical guarantees which are of a different nature from other approaches. Also, as presented, they do not support combinations of transformations. Methods such as DeepG [59] and PWL [21] cater for combined transformations but rely on computationally expensive techniques like DeepPoly [47] or VENUS [44], making them infeasible for large NNs due to scalability issues.

GeoRobust [22] investigates the worst-case combinations of transformations affecting a network's output for image inputs, by leveraging direct optimisation [66]. However, as shown in our main experiments, some unsound results have been revealed on the ImageNet dataset, which requires further analysis in terms of possible underestimations of the Lipschitz constant. Similar issues apply to existing verification methods based on global optimisation like DeepGO [18], as also shown in Table 10.

Algorithm 1 Geometric Robustness Validation of Neural Networks via $\mathrm{H}^2\mathrm{V}$.

Input: Original input $\boldsymbol{x}$ and its corresponding label y; N-dimensional perturbation variables of the geometric transformation $\boldsymbol{\theta} = [\gamma, t_{\text{horizontal}}, t_{\text{vertical}}, \lambda]$ with bounds $\boldsymbol{l}, \boldsymbol{b} \in \mathbb{R}^N$, Hilbert curve $h_{N,m}(\cdot)$, threat NN model; the property function $\tilde{f}$, the optimisation budget ϵ, and the maximum number of queries K.
Output: {UNSAT|SAT|Unknown}

$x_0 = 0, x_1 = 1, k = 2$
$z_j = \tilde{f}(x_j) = f(h(x_j)), j = 0, 1$ ▷ $h(x)$ is the m-approximation of the Hilbert curve
$\mathcal{O} = \{\}, \mathcal{I} = \{[0, 1]\}$ ▷ Initialisation for set of initial intervals
$\tilde{f}_m = \min\left\{\tilde{f}(a), \tilde{f}(b) \mid [a, b] \in \mathcal{I}\right\}$
if $\tilde{f}_m < 0$ **then**
  return SAT ▷ Misclassification
end if
while $k < K$ **do**
  for $i = [a, b] \in \mathcal{I}$ **do**
    Compute the estimation of the Hölder constant $\hat{H}_i$
    $s_i = \frac{b+a}{2} - \frac{\tilde{f}(b)-\tilde{f}(a)}{2\hat{H}_i(b-a)^{\frac{1-N}{N}}}$ ▷ Compute the intersection point for the interval
    $z_{\text{left}} = \tilde{f}(a) - \hat{H}_i(s_i - a)^{1/N}$ ▷ Compute the estimated lower bound from the left side
    $z_{\text{right}} = \tilde{f}(b) - \hat{H}_i(b - s_i)^{1/N}$ ▷ Compute the estimated lower bound from the right side
    $l_i = \min(z_{\text{left}}, z_{\text{right}})$
  end for
  Locate the interval from $i \in \mathcal{I}$, with $i = [a, b]$ who has minimum lower bound estimate l_i
  if $|b - a| \leq \epsilon$ **then**
    Compute the estimation of the lower bound of the function as $l_m = \min\{l_j \mid j \in \mathcal{I} \cup \mathcal{O}\}$.
    Compute the calibration of the lower bound $l_m \leftarrow l_m - \eta$
    if $l_m > 0$ **then**
      return UNSAT ▷ The property is validated to hold
    else
      return Unknown
    end if
  end if
  Compute $\tilde{f}(s_i)$, $k = k + 1$ ▷ Accept s_i as the new trial point
  $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i\} \cup \{[a, a'], [a', b]\}$, $\mathcal{O} \leftarrow \mathcal{O} \cup \{i\}$ ▷ Split the interval $[a, b]$ according to s_i
  Compute the current minimum of the function as $\tilde{f}_m = \min\left\{\tilde{f}(a), \tilde{f}(b) \mid [a, b] \in \mathcal{I}\right\}$,
  if $\tilde{f}_m < 0$ **then**
    return SAT ▷ Found a counterexample
  end if
end while
return Unknown

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.