Towards Verifying the Geometric Robustness of Large-Scale Neural Networks

Fu Wang[1], **Peipei Xu**[2], **Wenjie Ruan**[1*], **Xiaowei Huang**[2]

[1] Department of Computer Science, University of Exeter, Exeter, EX4 4QF, UK
[2] Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK
{fw377, w.ruan}@exeter.ac.uk, {peipei.xu, xiaowei.huang}@liverpool.ac.uk

Abstract

Deep neural networks (DNNs) are known to be vulnerable to adversarial geometric transformation. This paper aims to verify the robustness of large-scale DNNs against the combination of multiple geometric transformations with a provable guarantee. Given a set of transformations (e.g., rotation, scaling, etc.), we develop GeoRobust, a black-box robustness analyser built upon a novel global optimisation strategy, for locating the *worst-case* combination of transformations that affect and even alter a network's output. GeoRobust can provide *provable guarantees* on finding the *worst-case* combination based on recent advances in Lipschitzian theory. Due to its black-box nature, GeoRobust can be deployed on large-scale DNNs regardless of their architectures, activation functions, and the number of neurons. In practice, GeoRobust can locate the worst-case geometric transformation with high precision for the ResNet50 model on ImageNet in a few seconds on average. We examined 18 ImageNet classifiers, including the ResNet family and vision transformers, and found a positive correlation between the geometric robustness of the networks and the parameter numbers. We also observe that increasing the depth of DNN is more beneficial than increasing its width in terms of improving its geometric robustness. Our tool **GeoRobust** is available at https://github.com/TrustAI/GeoRobust.

Introduction

Although deep neural networks have achieved human-level performance, concerns are raised about their safety and reliability (Huang et al. 2020; Ruan et al. 2019; Wang et al. 2022; Ruan, Yi, and Huang 2021; Wu et al. 2020). In computer vision tasks, while deep learning models are known to be vulnerable to adversarial perturbations in pixel values (Szegedy et al. 2014; Croce and Hein 2020; Yin, Ruan, and Fieldsend 2022; Mu et al. 2021, 2022), Engstrom et al. (2019) show that a slight rotation of an input example can also fool DNNs. Although modern DNNs are believed to be able to learn geometric information from training data (Bakry et al. 2016), they are not yet invariant to simple adversarial geometric transformations (Zhang et al. 2020).

Although additive adversarial perturbation has received tremendous attention, geometric transformations are more common and applicable in the physical world but have been less studied (Zhang et al. 2023b). There is no efficient solution for searching the *worst-case* adversarial transformation with *provable guarantees* for *large-scale* DNNs. Engstrom et al. (2019) showed that, although adversarial geometric transformation can be discovered through the random pick, it is a highly non-convex task that gradient-ascent-based adversarial attacks perform poorly. Pei et al. (2017) adopt the exhaustive search to find the worst-case transformation that alters the target model's prediction, but its computational complexity grows exponentially with the dimension of the considered transformations. Some researchers have adopted verification techniques (Weng et al. 2018; Singh et al. 2019; Cohen, Rosenfeld, and Kolter 2019) to analyse geometric transformations.Relaxation-based approaches (Weng et al. 2018; Balunović et al. 2019; Mohapatra et al. 2020) require the L_p-norm based constraint on pixel space for an input example. However, as shown in Fig. 1, geometric transformation can significantly change the values of the pixels, leading to severe violence against the constraint in the pixel space while still preserving human imperceptibility. Therefore, the scalability of L_p norm-based verification is limited for dealing with geometric transformation. Recently, Fischer, Baader, and Vechev (2020); Li et al. (2021) showed that randomised smoothing could be utilised to verify robustness against a single geometric transform, but their methods cannot handle the combination of multiple geometric transformations. In addition, current methods can only provide a verifiable lower bound for verification purposes but cannot identify the *worst-case* geometric transformation that would actually minimise the model's confidence and potentially alter its prediction.

In this paper, we develop a novel black-box evaluation framework, GeoRobust, to study the geometric robustness, *i.e.*, the robustness of the model against adversarial geometric transformations. GeoRobust takes advantage of both recent developments in Lipschitzian optimisation methods (Jones, Perttunen, and Stuckman 1993; Gablonsky 2001) that provide provable guarantees on locating the worst-case transformation and the efficient parallel computation on Graphic Processing Units (GPUs). The workflow of GeoRobust is presented in Fig. 1. Given a set of geometric transformations and an input example, GeoRobust converges to the worst-case combination for minimising an

*Corresponding Author
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

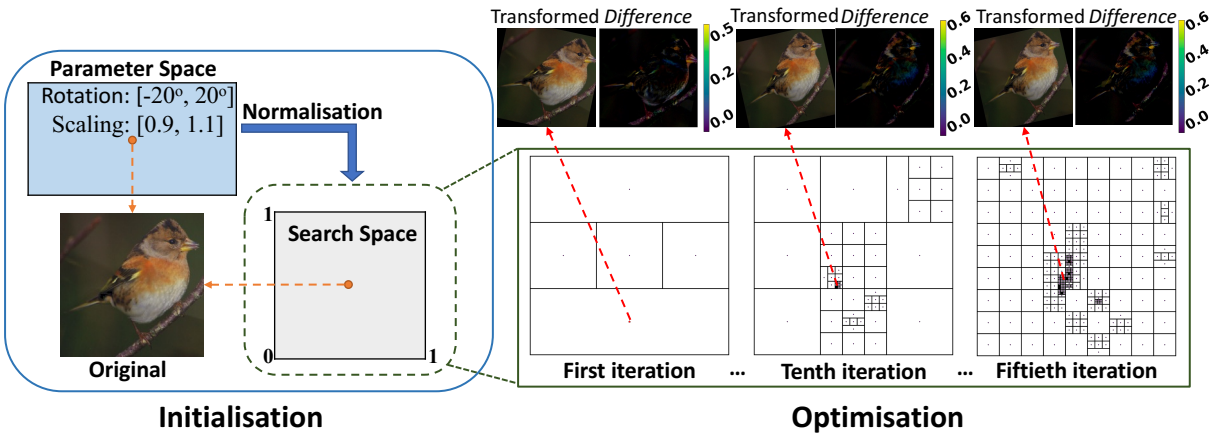


Figure 1: Schematic illustration of GeoRobust framework. After normalising the parameter space to a unit search space, GeoRobust performs a sequence of space divisions to find the global worst-case transformation.

adversarial loss within a finite number of queries. We enable GeoRobust to better utilise GPUs by easing its sampling condition. Besides, a lower bound estimation method is also introduced to make GeoRobust an *anytime* verification, which can produce the lower and upper bound of the worst-case loss value whenever the algorithm stops.

In summary, our key contributions lie in three aspects.

1. We prove the geometric transformation done by spatial transformation network (STN) (Jaderberg et al. 2015) is Lipschitz continuous. By stacking STN module in front of Lipschitz-continuous neural networks, we can analyse their geometric robustness with guaranteed convergence.
2. We develop GeoRobust, a black-box geometric robustness analyser, by taking advantage of Lipschitzian optimisation theory (Jones, Perttunen, and Stuckman 1993). The convergence of GeoRobust is theoretically guaranteed, and it is also highly efficient in practice. In our experiment, GeoRobust could find the worst-case adversarial transformations on an ImageNet image to evaluate a ResNet50 classifier with desirable precision in seconds.
3. We use GeoRobust to benchmark the geometric robustness of state-of-the-art ImageNet classifiers, including the ResNet family and vision transformers. There are two main takeaways from our experiments: *i)* the geometric robustness of DNNs has a positive correlation with the number of parameters; and *ii)* increasing the number of layers seems to be more effective than adding more hidden units in each layer in improving the geometric robustness of DNNs.

Preliminaries

Lipschitz continuity and Lipschitzian optimisation Previous studies indicate that the majority of modern DNNs are Lipschitz continuous (Szegedy et al. 2014; Ruan, Huang, and Kwiatkowska 2018; Virmaux and Scaman 2018; Zhang et al. 2023a). The Lipschitz constant for a DNN gives an upper bound on how fast its output could change when small perturbations are applied to its input (Szegedy et al. 2014; Ruan, Huang, and Kwiatkowska 2018). Such a concept is closely related to the robustness of DNN, but exactly computing the smallest Lipschitz constant of a DNN is proven to be an NP-hard problem (Virmaux and Scaman 2018).

Relying on the Lipschitz continuity, Lipschitzian optimisation is a query-based optimisation method that uses the Lipschitz constant of the objective function to gradually narrow the search space and locate the global optimum (Piyavskii 1972; Shubert 1972; Ruan, Huang, and Kwiatkowska 2018; Xu, Ruan, and Huang 2022; Zhang, Ruan, and Xu 2023). While the Lipschitz constant of the objective function is necessary for classic Lipschitzian optimisation, a novel Lipschitzian optimisation solution, DIRECT (Jones, Perttunen, and Stuckman 1993; Jones and Martins 2021a), does not require the Lipschitz constant to find the global optimum. As detailed in the methodology section, we improve the DIRECT method for studying the geometric robustness of DNNs.

Geometric transformations Geometric transformations are element-wise manipulation that can be conducted via several physically meaningful parameters (Szeliski 2022). Given an image example, $x \in \mathbb{R}^{H\times W\times C}$ with height H, width W, and colour channels C, the geometric transformation T_θ is carried out on each channel equally. Let $x_{\mathtt{c}} \in \mathbb{R}^{H\times W}$ be any channel of x and the output of T_θ be $x'_{\mathtt{c}}$. For a pixel in $x'_{\mathtt{c}}$ with index $(\mathtt{x}'_i, \mathtt{y}'_i)$, its value V_i is mapped to the pixel indexed by $(\mathtt{x}_j, \mathtt{y}_j)$ in $x_{\mathtt{c}}$ via a transformation matrix A_θ, *i.e.*,

$$\begin{bmatrix} \mathtt{x}_j \\ \mathtt{y}_j \end{bmatrix} = A_\theta \begin{bmatrix} \mathtt{x}'_i \\ \mathtt{y}'_i \\ 1 \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{bmatrix} \mathtt{x}'_i \\ \mathtt{y}'_i \\ 1 \end{bmatrix}. \quad (1)$$

We adopt Spatial Transformation Network (STN) (Jaderberg et al. 2015) to conduct the geometric transformation and use the bilinear sampling kernel to handle the projection of the non-integer index, which gives the transformation result:

$$V_i = \sum_h^H \sum_w^W U_{hw} \max\left(0, 1 - |\mathtt{x}_i - w|\right) \max\left(0, 1 - |\mathtt{y}_i - h|\right), \quad (2)$$

where U_{hw} denotes the value of a pixel, indexed by $(\mathtt{x}_h, \mathtt{y}_w)$, in $x_\mathtt{c}$, and the index does not need to be an integer.

Problem Formulation

Given a neural network $F : \mathbb{R}^N \to \mathbb{R}^K$, an input example $x \in \mathbb{R}^N$, and its label $y \in \{1, \ldots, K\}$, we aim to find the optimal combination of several geometric transformations T_θ that can minimise $\ell : \mathbb{R}^n \to \mathbb{R}$, *i.e.*,

$$\min_{\theta \in \Theta} \ell(\theta; F, x, y), \tag{3}$$

where Θ is the adversarial space that contains all feasible θ, and ℓ denotes the margin loss defined as

$$\ell(\theta; F, x, y) = F_y(T_\theta(x)) - \max_{k \in \{1,\ldots,K\} \setminus \{y\}} F_k(T_\theta(x)), \tag{4}$$

which allow us to determine whether the model F can be fooled by $T_\theta(x)$ via verifying the lower bound of ℓ. Specifically, if $\inf_{\theta \in \Theta} \ell(\theta; F, x, y) > 0$ is satisfied, the robustness of the model F on the example x would be verified.

Regarding geometric transformation, we consider rotation, translation, and isotropic scaling. The corresponding transformation matrix A_θ can be written as

$$A_\theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} = \begin{bmatrix} \lambda \cos\gamma & -\sin\gamma & t^{hor} \\ \sin\gamma & \lambda\cos\gamma & t^{vrt} \end{bmatrix}, \tag{5}$$

where λ is the scaling factor, γ is the rotation angle, and t^{hor} and t^{vrt} control the horizontal and vertical translation.

Methodology

This section introduces a Lipschitzian optimisation-based approach, GeoRobust, to search for the worst-case transformation. GeoRobust is composed of four components: *1)* a Lipschitz continuous module for performing geometric transformations; *2)* a space division procedure; *3)* a mechanism to select Potential Optimal (PO) subspaces that are more likely to contain the global minimum points than others; *4)* and an anytime estimation of the global minimum. While the space division procedure and the PO subspace selection are adopted from the DIRECT algorithm, we extend the definition of PO subspace and encourage the algorithm to query more subspaces at each iteration. Because all evaluations at the same iteration can be done parallelly in a single forward propagation, our method could reach convergence within reduced iterations. Furthermore, GeoRobust can estimate the model's Lipschitz constant and produce a reasonable lower bound of the given loss function. The pseudocode of GeoRobust and related proofs are provided in **Appendix**[1].

Notations Given a matrix $P \in \mathbb{R}^{2\times n}$, one can define an n-dimensional parameter space, in which the upper and lower bounds on i-th transformation factors are given by P_i, where $i \in \{1, \ldots, n\}$. GeoRobust normalises the parameter space into an n-dimensional unit hypercube, namely the search space, whose centre point corresponds to the identical transformation. For a hyperrectangle with index q, denoted by H_q, in the search space, the value of the objective function at its centre point c_q is denoted by $\ell(c_q)$. We denote by l_i^q the side length w.r.t. the i-th dimension, and by $\boldsymbol{e}_i$ the unit vector along i-th dimension. The size of H_q is defined as $\sigma_q = \max_{i\in\{1,\ldots,n\}} \frac{1}{2} l_i^q$, which is the same L_∞ norm based measurement used by Gablonsky (2001). For all hyperrectangles within the search space, we denote by $\mathcal{H}$ the set of hyperrectangles' indexes, and by $\ell_{\min} = \min_{p\in\mathcal{H}} \ell(c_p)$ the current best query result. The Lipschitz constant of the objective function w.r.t. the search space is denoted by $\tilde{K}$. GeoRobust computes the slope $\hat{K}$ between queried points w.r.t. the parameter space during optimisation and produces $\ell^*_{\min}$, an estimation of the lower bound of $\ell_{\min}$.

[1] Available at https://github.com/TrustAI/GeoRobust.

Geometric Transformations Module

The convergence guarantee of GeoRobust is related to the Lipschitz continuity of the target model. We give the following lemma to show that geometric transformations with bilinear sampling kernel are Lipschitz continuous, which means, as long as a DNN model is Lipschitz continuous, stacking a geometric transformation module in front of it would not compromise the Lipschitz continuity (Tsuzuku, Sato, and Sugiyama 2018).

Lemma 1. *Given an input image example $x \in \mathbb{R}^{H\times W\times C}$ and the ranges of transformation factors, the first-order derivative of geometric transformation with bilinear sampling w.r.t. each transformation factor is bounded.*

Finding Optimal Geometric Transformation

GeoRobust first divides the search space into subspaces according to the query results at their centre points. Then some subspaces that are more likely to contain the global minimum than others will be chosen as PO subspaces. GeoRobust separates selected PO spaces and identifies new ones throughout each iteration of the optimisation process till the termination criteria are satisfied.

Space division As the only hypercube after initialisation, the initial PO subspace is the united search space itself. GeoRobust **trisects** the subspace and assigns the generated new subspace according to the query result, where the larger hyperrectangles include the better query result. Without loss of generality, let H_p be a PO subspace, which is an n-dimensional hyperrectangle containing m dimensions with long sides of a length 3^{-d}, where $m \leq n$, and $n-m$ dimensions with short sides of a length 3^{-d-1}. GeoRobust ignores short sides and queries the value of the object function at the points $c \pm 3^{-d-1}\boldsymbol{e}_i$, where $i \in \{1, \ldots, m\}$. For each dimension with long sides, the best query result is given by

$$w_i = \min\left(\ell(c + 3^{-d-1}\boldsymbol{e}_i), \ell(c - 3^{-d-1}\boldsymbol{e}_i)\right). \tag{6}$$

As GeoRobust performs trisection only during division, the sizes of new subspaces are deterministic. By dividing the above H_p, GeoRobust creates $2m + 1$ new subspaces, including 3 sub-hypercubes with the side length of 3^{-d-1} and $m-1$ pairs of hyperrectangles, which have 1 to $m-1$ dimensions with long sides of length 3^{-d}. The point corresponding to the best query result, $\min_{i\in\{1,\ldots,m\}} w_i$, is a centre point

of a hyperrectangle with $m-1$ long sides. This space division procedure is visualised in **Appendix**. Overall, such a division strategy encourages GeoRobust to divide the search space uniformly and further explore the area around the current best result, which we will detail later in the PO subspace selection.

Identifying potential optimal subspaces The space division procedure creates new subspaces, and the next step is to locate new PO subspaces for further division. Ideally, a PO hyperrectangle H_p is expected to satisfy two conditions (Jones, Perttunen, and Stuckman 1993):

$$\ell(c_p) - \tilde{K}\sigma_p \leq \ell(c_q) - \tilde{K}\sigma_q, \forall q \in \mathcal{H}, \tag{7}$$

$$\ell(c_p) - \tilde{K}\sigma_p \leq \ell_{\min} - \tau|\ell_{\min}|. \tag{8}$$

Inequation (7) indicates that only the hyperrectangles with the potential to improve the current $\ell_{\min}$ can be chosen as PO subspaces. Meanwhile, the second condition (8) ensures that the possible improvement in the chosen subspaces is greater than $\tau|\ell_{\min}|$, where a reasonable choice of τ is between 10^{-3} and 10^{-7} (Jones and Martins 2021b). Taking advantage of DIRECT optimisation, GeoRobust does not need to know the Lipschitz constant. The following lemma demonstrates how to search for PO hyperrectangles in the absence of $\tilde{K}$.

Lemma 2. *(Gablonsky 2001) Given the index set $\mathcal{H}$ and a positive tolerance $\tau > 0$. Let $\ell_{\min}$ denote the current best query result. Let $\mathcal{H}_1^p = \{q \in \mathcal{H} : \sigma_q < \sigma_p\}$, $\mathcal{H}_2^p = \{q \in \mathcal{H} : \sigma_q > \sigma_p\}$ and $\mathcal{H}_3^p = \{q \in \mathcal{H} : \sigma_q = \sigma_p\}$. A hyperrectangle H_p is said to be potentially optimal if*

$$\ell(c_p) \leq \ell(c_q), \forall q \in \mathcal{H}_3^p, \tag{9}$$

and there is a $\tilde{K} > 0$ such that

$$\max_{q\in\mathcal{H}_1^p} \frac{\ell(c_p)-\ell(c_q)}{\sigma_p-\sigma_q} \leq \tilde{K} \leq \min_{q\in\mathcal{H}_2^p} \frac{\ell(c_q)-\ell(c_p)}{\sigma_q-\sigma_p}, \tag{10}$$

and

$$\begin{cases} \tau \leq \frac{\ell_{\min}-\ell(c_p)}{|\ell_{\min}|} + \frac{\sigma_p}{|\ell_{\min}|}\min_{q\in\mathcal{H}_2^p}\frac{\ell(c_q)-\ell(c_p)}{\sigma_q-\sigma_p}, & \text{if } \ell_{\min} \neq 0, \\ \ell(c_p) \leq \sigma_p \min_{q\in\mathcal{H}_2^p}\frac{\ell(c_q)-\ell(c_p)}{\sigma_q-\sigma_p}, & \text{otherwise.} \end{cases} \tag{11}$$

Generalising PO Conditions to Unleash the Power of Parallel Computation

The conditions in Lemma 2 are meant to select a small number of subspaces to reduce the total number of function evaluations, which is typically the most time-consuming procedure. However, by leveraging modern deep learning frameworks, one can easily query multiple examples on a target DNN via a single forward propagation on GPUs, where the computational time difference between evaluating a single sample and a batch of samples is marginal. Therefore, relaxing the constraint given by inequation (9), we define the following α candidate set to select more subspaces.

Definition 1 (α candidate set)**.** Following the same notions in Lemma 2, we define the α candidate set as

$$\mathcal{H}_\alpha = \begin{cases} \varnothing, & \text{if } \max_{p\in\mathcal{H}_3^p} s_p \leq 0, \\ \{p_1, \ldots, p_{\alpha'} : \max \sum_{j=1}^{\alpha'} s_{p_j}\}, & \text{otherwise,} \end{cases} \tag{12}$$

where $\alpha' \leq \alpha$ and the optimal score s_p is given by

$$s_p = \min_{q\in\mathcal{H}_2^p} \frac{\ell(c_q)-\ell(c_p)}{\sigma_q-\sigma_p} - \max_{q\in\mathcal{H}_1^p} \frac{\ell(c_p)-\ell(c_q)}{\sigma_p-\sigma_q}. \tag{13}$$

Modifying the condition (10) into the score described in Eq. (13) allows us to rank the potential minimum contained by subspaces with the same size. The proposed α candidate set is easy to control. When $\alpha = 1$, Definition 1 degrades to condition (10), while increasing α, GeoRobust explores more subspaces in each iteration. As described in the next section, all subspaces will be subdivided by GeoRobust after a certain number of iterations. Instead of only partitioning spaces satisfying inequation (7), α candidate set also selects hyperrectangles that would likely satisfy Lemma 2 in the next few rounds. While the number of queries would rise, using the α candidate set enables GeoRobust to discover the optimal subspace quickly. On the other hand, because the function evaluations can be done in parallel on GPUs, replacing condition (10) with α candidate set only has a small influence on the computational time cost.

Stop Criteria and Convergence Analysis

In practice, GeoRobust is limited by three factors: the maximal number of iteration T, the maximal number of queries Q, and the maximal number of trisection along each dimension, which is denoted by depth D. The first two stop criteria are straightforward. We stop the optimisation once the computational budget runs out. The limitation on depth is applied for two reasons (Gablonsky 2001). On the one hand, it puts a limitation on the smallest size of subspaces. By doing so, GeoRobust is compelled to halt local search and encouraged to conduct global exploration when the current optimal subspace is sufficiently small, which accelerates the convergence. On the other hand, defining the smallest subspace size also sets an upper bound for the number of queries. For an n-dimensional search space, GeoRobust could conduct up to 3^{nD} times of queries, which is equivalent to a grid search. Besides, our implementation adopted the L_∞ norm to measure hyperrectangles' size (Gablonsky 2001). Such a measurement simplifies both space division and PO space selection. For the former, the L_∞ norm is more computationally efficient than the Euclidean norm (Jones and Martins 2021b). For the latter, hyperrectangles are grouped by their longest side length under the L_∞ norm, which introduces less number of different sizes to consider.

Convergence analysis GeoRobust is guaranteed to converge to the global minimum within a pre-defined small tolerance after a finite number of queries if the objective function is continuous (Jones, Perttunen, and Stuckman 1993). This guarantee comes from the following observation shown in Remark 1.

Remark 1. *(Gablonsky 2001) Following the same notions in Lemma 2, there is at least one hyperrectangle H_p will be identified as PO subspace at each iteration, where H_p satisfies $\mathcal{H}_2^p = \varnothing$ and makes inequation* (9) *holds so that every hyperrectangle will be subdivided after finite iterations.*

Note that theoretically proving a specific DNN is Lipschitz continuous is beyond the scope of this paper, and ex-

isting works demonstrate the Lipschitz continuity of convolutional neural networks (Ruan, Huang, and Kwiatkowska 2018) and vision transformers (Vuckovic, Baratin, and des Combes 2021; Wang and Ruan 2022; Wang, Ruan, and Yin 2023) used in the image classification task. In addition, given by Lemma 1, we prove that the geometric transformation is Lipschitz continuous. So, as long as the neural network satisfies a Lipschitz condition, the objective function (3) is Lipschitz continuous, and GeoRobust is guaranteed to locate to the global minimum after a sufficient number of queries. The convergence complexity is described in Theorem 1.

Theorem 1. *Let C be the n-dimensional united search space and $\tilde{K}$ be the Lipschitz constant of ℓ w.r.t. C. The gap between current minima and global minima after T iterations can be written as*

$$\ell_{\min} - \min_{c \in C} \ell(c) \leq \varepsilon < \tilde{K} \cdot (T+1)^{-\frac{1}{n}}. \tag{14}$$

Therefore, to achieve any desired ε, we need up to $\mathcal{O}\big((\tilde{K}/\varepsilon)^n\big)$ iterations.

Estimating the Global Minimum

While GeoRobust is guaranteed to find the global minimum eventually, we enable it to be an anytime analyser that can utilise intermediate query results to estimate the lower bound of the global minimum at each iteration.

Recall that the parameter space is defined via the matrix $P \in \mathbb{R}^{2\times n}$ that contains the upper and lower bounds of n transformation factors. To divide m dimensions of a hyperrectangle, GeoRobust samples and evaluates new points at $c \pm 3^{-1} \cdot l_i \boldsymbol{e}_i$, where $i \in \{1, \ldots, m\}$. The slopes along the i-th dimension are given by

$$\hat{K}_{c_i^+} = \frac{\|\ell(c) - \ell(c_i^+)\|}{d_i^c} \quad \text{and} \quad \hat{K}_{c_i^-} = \frac{\|\ell(c) - \ell(c_i^-)\|}{d_i^c},$$

where c_i^+ and c_i^- are short hands for $c \pm 3^{-1} \cdot l_i \boldsymbol{e}_i$, and we denote by $d_i^c = 3^{-1} \cdot l_i P_i \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ the distance between c and $c_i^{+/-}$ within the parameter space. Then $\hat{K}_c$, the local slope of c, is updated to be the largest local slope, *i.e.*,

$$\hat{K}_c = \max\{\hat{K}_{c_1^+}, \hat{K}_{c_1^-}, \ldots, \hat{K}_{c_m^+}, \hat{K}_{c_m^-}\}.$$

GeoRobust updates the local slope after space division so that it can estimate the global minimum based on the query results at that time. Let c_o denote the centre of the current optimal hyperrectangle H_o that has $\ell(c_o) = \ell_{\min}$ and $\hat{K} = \max_{q \in \mathcal{H}} \hat{K}_{c_q}$, the lower bound of global minimum can be estimated via

$$\ell^*_{\min} = \ell(c_o) - \hat{K}_{\max} \bar{\sigma}_o, \tag{15}$$

where we take a relaxation on σ_o and compute it via the Manhattan distance, *i.e.*, $\bar{\sigma}_o = \frac{1}{2}\sum_{i=1}^n d_i^o$. Therefore, as long as GeoRobust can locate a H_o containing the global minimum $\hat{\ell}_{\min}$, we have $\ell^*_{\min} \leq \hat{\ell}_{\min}$.

Experiments

Our experiments include three parts. First, we compare GeoRobust to state-of-the-art baseline methods for verifying robustness against three geometric transformations, *i.e.*, rotation, translation, and scaling. Then, we take advantage of GeoRobust to efficiently benchmark popular large-scale networks on ImageNet regarding their robustness over the combination of all three transformations. Finally, we conduct an empirical analysis to study the impact of the depth and α conditions on the convergence of GeoRobust.

General setup For geometric transformations, we denoted by $R(\gamma)$ the rotation angle between $\pm\gamma$, by $S(\lambda)$ the scaling range between $1 \pm \lambda$, and by $T(t^{hor}, t^{vrt})$ the translation that moves an example up to t^{hor} and t^{vrt} pixels horizontally and vertically, respectively. For the model architectures, the MNIST classifier is a network with four convolutional layers and three linear layers, and the CIFAR10 classifier's architecture is ResNet101. We adopted the pre-trained MNIST and CIFAR10 models released by Li et al. (2021) and utilised TIMM, a model zoo of ImageNet classifiers, to investigate the geometric robustness of popular large-scale DNNs. Our experiment is performed on a machine with an Intel i7-10700KF CPU, an RTX 3090 GPU, and 48 gigabytes of memory. More implementation details can be found in **Appendix**.

Comparison with Previous Works

Since GeoRobust only requires queries on the target models' output, it can be easily deployed on any pretrained neural network. We follow the same setup used in TSS (Li et al. 2021) and GSmooth (Hao et al. 2022) and apply GeoRobust on their pretrained models to conduct robustness verification on MNIST and CIFAR10. For GeoRobust, the robustness of an example is verified if the corresponding lower bound $\ell^*_{\min} > 0$. Because exhaustive search is computationally infeasible, we implement a grid search with a sufficient computational budget to run through the parameter space to test whether the model's prediction on each input example can be altered, which serves as the ground truth on the verified accuracy. Please note that GeoRobust works on L_∞-norm based parameter space, while Li et al. (2021) uses L_2-norm based constraint on the translation, which is currently inapplicable for GeoRobust. Therefore, the comparison is done on rotation and scaling transformations. Although DeepG (Balunović et al. 2019) and TSS (Li et al. 2021) can analyse the geometric robustness of ImageNet classifiers, they are time-consuming and inefficient when dealing with transformation combinations. Besides, we failed to properly reload the ImageNet classifiers evaluated by TSS (see **Appendix** for details). Therefore, the evaluation on ImageNet is done on a ResNet50 model, the same architecture used by Li et al. (2021), against different combinations of transformations, and we compare the performance with grid search and random pick.

The comparison results on MNIST and CIFAR10 are summarised in Tab. 1, in which we report the verified accuracy determined by each baseline method. GeoRobust outperforms previous methods under all scenarios and reports

Dataset	Trans.	GeoRobust	Gsmooth	TSS	DeepG	Interval	Semantify-NN	DistSPT	TSS attack	Grid Search
MNIST	$R(50°)$	**98.2%**	95.7%	97.4%	$\leq$ 85.8%	$\leq$6.0%	$\leq$ 92.48%	82%	98.2%	98.2%
	$S(0.3)$	**99.2%**	95.9%	97.2%	85.0%	16.4%	-	-	99.2%	99.2%
CIFAR10	$R(10°)$	**74.8%**	65.6%	70.6%	62.5%	20.2%	-	37%	76.4%	74.8%
	$R(30°)$	**66.4%**	-	63.6%	10.6%	0.0%	49.37%	22%	69.4%	66.4%
	$S(0.3)$	**63.4%**	54.3%	58.8%	0.0%	0.0%	-	-	67.0%	63.4%

Table 1: Comparing with baseline methods on MNIST and CIFAR-10 against rotation and Scaling. We denote by $-$ an unsupported setting and by 0% a failed verification. Baselines' performance is adopted from (Li et al. 2021; Hao et al. 2022).

Methods	Transformation			
	R	T	$S+T$	$R+T+S$
GeoRobust	58%	57%	57%	46%
Random pick	58%	59%	60%	49%
Grid search	58%	59%	57%	46%

Table 2: Verify geometric robustness on ImageNet with ResNet50 model, whose vanilla accuracy is 74%, against $R(20°)$, $S(0.1)$, and $T(22.4, 22.4)$ transformations.

the same verified accuracy as grid search. The experiment demonstrates the effectiveness of GeoRobust in verifying the geometric robustness against 1-dimensional transformation. The evaluation on ImageNet is summarised in Tab. 2, in which we can see that the accuracy verified by GeoRobust is comparable to or better than the grid search. Random pick with sufficient queries can achieve a similar performance as grid search. Still, it tends to perform worse as the dimension of parameter space becomes larger. In addition, as the minimum found by grid search is more likely to be the ground truth minimum, we mark an example as a match if its corresponding minimum found by a method is equal to or smaller than the minimum found by grid search. As shown in Fig. 2, we can see that the estimated lower bound achieves considerable precision with a limited number of queries. The performance of only verifying the translation is slightly worse than other transformations, where the reason might be the distortion introduced by bilinear sampling.

Runtime The effectiveness of GeoRobust is highly related to the transformation's dimensions. In Tab. 1, GeoRobust is only performed on 1-dimensional transformation. Its average runtime is 0.18 seconds and 0.72 seconds per example on MNIST and CIFAR10, respectively. Furthermore, the average runtime for analysing the ResNet50 ImageNet classifier from 1-dimensional to 4-dimensional transformations are 2.4 seconds, 3.6 seconds, 4.0 seconds, and 4.5 seconds. In comparison, according to (Li et al. 2021), it takes TSS 17.7 and 1201.2 seconds, respectively, to analyse an MNIST example and an ImageNet example on the same model architectures w.r.t. a 1-dimensional transformation.

Benchmarking Geometric Robustness

This section investigates the robustness of large-scale DNN classifiers against geometric transformations. Since GeoRobust can efficiently locate a worst-case combination of transformations in a black-box manner, we utilise GeoRobust to evaluate the geometric robustness of large-scale ImageNet classifiers against the combination of rotation $R(20°)$, translation $T(22.4, 22.4)$, and scaling $S(0.1)$.

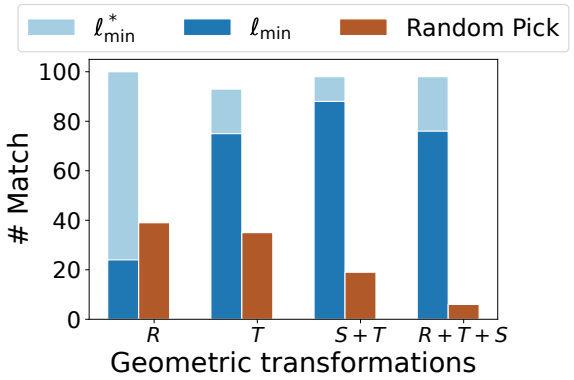
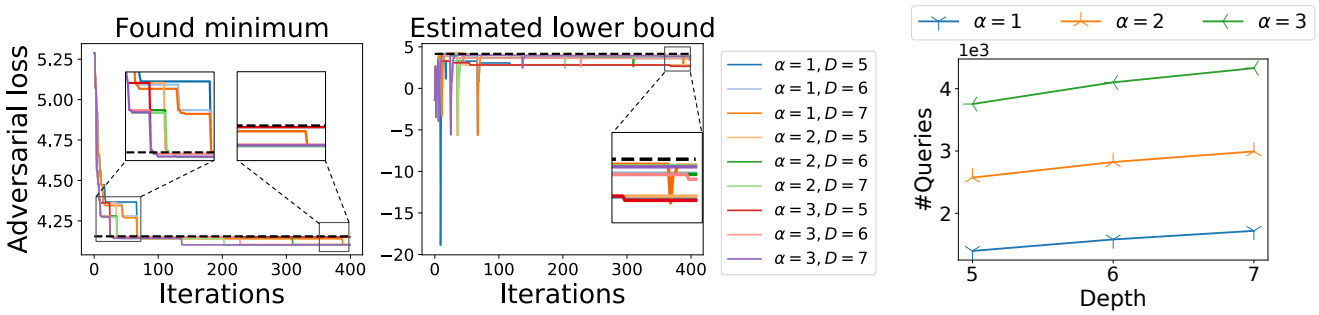


Figure 2: Comparing the global minimum found by grid search, random pick, and GeoRobust. The geometric transformations are the same as in Tab. 2. We mark an example as a match if its corresponding minimum found by a method is equal to or smaller than the minimum found by grid search.

Models	Vanilla	Attack	Verified	#Parameters
Inception v3.	73.60%	28.20%	24.20%	2.4×10^7
Inception $v3_{adv}$	75.00%	30.60%	27.00%	2.4×10^7
Inception v4	78.40%	40.20%	36.40%	4.3×10^7
ResNet34	64.40%	10.60%	9.00%	2.2×10^7
ResNet50	78.40%	54.00%	31.12%	2.6×10^7
WideResNet50	81.60%	49.40%	40.00%	6.9×10^7
ResNet101	80.00%	54.20%	48.20%	4.5×10^7
ResNet152	79.40%	53.80%	46.20%	6.0×10^7
Vit_{32}	75.60%	23.40%	19.00%	8.8×10^7
Vit_{16}	81.40%	41.20%	34.20%	8.6×10^7
Large Vit_{16}	83.40%	49.20%	40.20%	3.0×10^8
$Beit_{16}$.	83.80%	56.40%	52.00%	6.5×10^7
Large $Beit_{16}$	**85.60%**	**65.60%**	**58.20%**	2.3×10^8
Gmlp	77.96%	40.80%	36.80%	1.9×10^7
Mixer	72.20%	27.20%	23.40%	6.0×10^7
Swin	80.20%	34.60%	13.20%	8.8×10^7
Xcit	76.80%	40.40%	20.60%	8.4×10^7
Pit	79.40%	36.60%	20.00%	7.4×10^7

Table 3: Benchmarking the geometric robustness of eighteen ImageNet classifiers.



(a) (b)

Figure 3: Carrying out GeoRobust with different combinations of candidates set size α and depth D on ResNet50. The black dot line in 3(a) corresponds to a global minimum found in a grid search with 2.5×10^5 function evaluations.

From Tab. 3, we can see that *1)* models with more parameters appear to have better geometric robustness than those with fewer; *2)* widening a network seems less beneficial than deepening it in terms of improving the geometric robustness; *3)* the large version of Beit showed the best geometric robustness, whereas the basic Beit model is the second most robust model. This phenomenon suggests that bidirectional modelling could be helpful for DNNs learning geometric information and obtaining geometric robustness; *4)* comparing the performance between Inception V3 and Inception V3$_{adv}$, an adversarially trained model, we can see that adversarial training does not significantly improve the model's geometric robustness.

Empirical Analysis

In Fig. 3, we carry out GeoRobust with different combinations of candidates set size α and depth D on ResNet50. Increasing the size of α candidates set enables GeoRobust to be more efficient in exploring the search space and locating the optimal subspace. Due to the limitation on the subspaces' minimal size, as the depth gets larger, the optimal transformation combination found by GeoRobust is closer to the ground truth worst-case, and the estimated lower bound is closer to the global minimum as well. It can be observed that the upper bound remains unchanged after convergence, while the estimated lower bound would be updated whenever GeoRobust finds a larger local slope, which is why the estimations change in the right side plot of Fig. 3(a). As shown in Fig. 3(b), while the impact of depth on computational cost is trivial, increasing the α candidates set would significantly raise the total number of function queries in fixed iterations. The runtime of GeoRobust with $\alpha = 1$ and $D = 5$ is 6.9 seconds, and the runtime is 16.1 seconds when it is carried out at $\alpha = 3$ and $D = 7$. We can see that the runtime increases sub-linearly with the number of queries because the queries are done parallelly on GPUs.

Related Works

In this paper, we compared GeoRobust to Interval (Singh et al. 2019), DeepG (Balunović et al. 2019), Semantify-NN (Mohapatra et al. 2020), TSS (Li et al. 2021), GSmooth (Hao et al. 2022), and DistSPT (Fischer, Baader, and Vechev 2020). DeepG (Balunović et al. 2019), Semantify-NN (Mohapatra et al. 2020), and Interval extend verification techniques designed for L_p-norm based additive perturbation. Both Semantify-NN (Mohapatra et al. 2020) and GSmooth (Hao et al. 2022) introduce small networks to simulate the geometric manipulation, where Semantify-NN adopts a linear relaxation-based verification (Weng et al. 2018) and GSmooth applies random smoothing. DistSPT (Fischer, Baader, and Vechev 2020) and TSS (Li et al. 2021) are also randomised smoothing based approaches, where TSS is a black-box analyser that is scalable to large DNNs. Besides, although the parameter space of control factors for most geometric manipulations is continuous, the image pixels' coordinates are bounded integers, which means the possible outcomes for a particular set of transformations are finite. Pei *et al.* (Pei et al. 2017) empirically evaluated the robustness of DNNs against geometric transformations by enumerating all possible values. In contrast, our GeoRobust is a query-based black-box analyser that is fundamentally different to the above methods. We demonstrated that as long as the target model is Lipschitz continuous, GeoRobust can verify the robustness of large-scale DNNs against a combination of geometric transformations in seconds. The combination with probabilistic approaches (Zhang, Ruan, and Fieldsend 2022) will be explored in our future works.

Conclusion

In this paper, we propose a black-box analyser, GeoRobust, to efficiently verify the robustness of large-scale DNNs against geometric transformation. Given the ranges of multiple geometric transformations, GeoRobust can find the worst-case manipulation that can minimise an adversarial loss without knowing the internal structures of the target model. Theoretically, we prove the Lipschitz continuity of geometric transformations operated by STN and analyse the convergence complexity of GeoRobust. On the methodology side, we generalise the sampling strategy to better leverage GPU parallel computation and design an anytime estimation method to approximate the lower bound. With GeoRobust, we systematically benchmark the geometric robustness of popular ImageNet classifiers. Our empirical study shows that larger neural networks are more robust against geometric manipulation. Deepening a network improves its geometric robustness better than increasing its width.

Acknowledgements

This work is supported by Partnership Resource Fund of ORCA Hub via the UK EPSRC under project [EP/R026173/1]. XH has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 956123, and is also supported by the UK EPSRC under project [EP/T026995/1]. FW is funded by the Faculty of Environment, Science and Economy at the University of Exeter. PX is funded by the department of computer science at the University of Liverpool. PX contributed to this work equally and conducted this work while she was visiting the University of Exeter. We would like to thank Haozhe Wang, Anjan Dutta, and the anonymous reviewers for their helpful comments and Linyi Li for sharing the pretrained models with us.

References

Bakry, A.; Elhoseiny, M.; El-Gaaly, T.; and Elgammal, A. M. 2016. Digging Deep into the Layers of CNNs: In Search of How CNNs Achieve View Invariance. In *ICLR*.

Balunović, M.; Baader, M.; Singh, G.; Gehr, T.; and Vechev, M. T. 2019. Certifying Geometric Robustness of Neural Networks. In *NeurIPS*.

Cohen, J. M.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *ICML*.

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.

Engstrom, L.; Tran, B.; Tsipras, D.; Schmidt, L.; and Madry, A. 2019. Exploring the Landscape of Spatial Robustness. In *ICML*.

Fischer, M.; Baader, M.; and Vechev, M. T. 2020. Certified Defense to Image Transformations via Randomized Smoothing. In *NeurIPS*.

Gablonsky, J. M. X. 2001. *Modifications of the DIRECT algorithm*. North Carolina state university.

Hao, Z.; Ying, C.; Dong, Y.; et al. 2022. GSmooth: Certified Robustness against Semantic Transformations via Generalized Randomized Smoothing. In *ICML*.

Huang, X.; Kroening, D.; Ruan, W.; Sharp, J.; Sun, Y.; Thamo, E.; Wu, M.; and Yi, X. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37: 100270.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; and Kavukcuoglu, K. 2015. Spatial Transformer Networks. In *NeurIPS*.

Jones, D. R.; and Martins, J. R. 2021a. The DIRECT algorithm: 25 years Later. *Journal of Global Optimization*, 79(3): 521–566.

Jones, D. R.; and Martins, J. R. R. A. 2021b. The DIRECT algorithm: 25 years Later. *J. Glob. Optim.*, 79(3): 521–566.

Jones, D. R.; Perttunen, C. D.; and Stuckman, B. E. 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79: 157–181.

Li, L.; Weber, M.; Xu, X.; et al. 2021. TSS: Transformation-Specific Smoothing for Robustness Certification. In *ACM Conference on Computer and Communications Security*.

Mohapatra, J.; Weng, T.-W.; Chen, P.-Y.; Liu, S.; and Daniel, L. 2020. Towards Verifying Robustness of Neural Networks Against A Family of Semantic Perturbations. In *CVPR*.

Mu, R.; Ruan, W.; Marcolino, L. S.; and Ni, Q. 2022. 3DVerifier: efficient robustness verification for 3D point cloud models. *Machine Learning*, 1–28.

Mu, R.; Ruan, W.; Soriano Marcolino, L.; and Ni, Q. 2021. Sparse Adversarial Video Attacks with Spatial Transformations. In *BMVC*.

Pei, K.; Cao, Y.; Yang, J.; and Jana, S. 2017. Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems. *arXiv*, abs/1712.01785.

Piyavskii, S. 1972. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4): 57–67.

Ruan, W.; Huang, X.; and Kwiatkowska, M. 2018. Reachability analysis of deep neural networks with provable guarantees. In *IJCAI*.

Ruan, W.; Wu, M.; Sun, Y.; Huang, X.; Kroening, D.; and Kwiatkowska, M. 2019. Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance. In *IJCAI*.

Ruan, W.; Yi, X.; and Huang, X. 2021. Adversarial Robustness of Deep Learning: Theory, Algorithms, and Applications. In *CIKM*.

Shubert, B. O. 1972. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3): 379–388.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. T. 2019. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL): 41:1–41:30.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; et al. 2014. Intriguing properties of neural networks. In *ICLR*.

Szeliski, R. 2022. *Computer vision: algorithms and applications*. Springer Nature.

Tsuzuku, Y.; Sato, I.; and Sugiyama, M. 2018. Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks. In *NeurIPS*.

Virmaux, A.; and Scaman, K. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *NeurIPS*.

Vuckovic, J.; Baratin, A.; and des Combes, R. T. 2021. On the Regularity of Attention. *arXiv*, abs/2102.05628.

Wang, F.; Zhang, C.; Xu, P.; and Ruan, W. 2022. *Deep learning and its adversarial robustness: A brief introduction*, 547–584. World Scientific.

Wang, Z.; and Ruan, W. 2022. Understanding Adversarial Robustness of Vision Transformers via Cauchy Problem. In *ECML/PKDD*.

Wang, Z.; Ruan, W.; and Yin, X. 2023. ODE4ViTRobustness: A tool for understanding adversarial robustness of Vision Transformers. *Software Impacts*, 15: 100449.

Weng, T.-W.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.-J.; Boning, D.; Dhillon, I. S.; and Daniel, L. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *ICML*.

Wu, M.; Wicker, M.; Ruan, W.; Huang, X.; and Kwiatkowska, M. 2020. A game-based approximate verification of deep neural networks with provable guarantees. *Theoretical Computer Science*, 807: 298–329.

Xu, P.; Ruan, W.; and Huang, X. 2022. Quantifying safety risks of deep neural networks. *Complex & Intelligent Systems*, 1–18.

Yin, X.; Ruan, W.; and Fieldsend, J. 2022. DIMBA: discretely masked black-box attack in single object tracking. *Machine Learning*, 1–19.

Zhang, C.; Ruan, W.; Wang, F.; Xu, P.; Min, G.; and Huang, X. 2023a. Model-Agnostic Reachability Analysis on Deep Neural Networks. In *PAKDD*.

Zhang, C.; Ruan, W.; and Xu, P. 2023. Reachability Analysis of Neural Network Control Systems. In *AAAI*.

Zhang, T.; Ruan, W.; and Fieldsend, J. E. 2022. PRoA: A Probabilistic Robustness Assessment against Functional Perturbations. In *ECML/PKDD*.

Zhang, Y.; Ruan, W.; Wang, F.; and Huang, X. 2020. Generalizing universal adversarial attacks beyond additive perturbations. In *ICDM*.

Zhang, Y.; Ruan, W.; Wang, F.; and Huang, X. 2023b. Generalizing Universal Adversarial Perturbations for Deep Neural Networks. *Machine Learning*.