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
† Corresponding Author

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). 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. 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 purpose 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, called GeoRobust, to study the geometric robustness, *i.e.*, the robustness of the model against adversarial geometric transformations. GeoRobust takes advantage of both recent development in Lipschitzian optimisation methods (Jones, Perttunen, and Stuckman 1993; Gablonsky 2001) that provides 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 geo-

[*] This work was done when Peipei was visiting the University of Exeter.

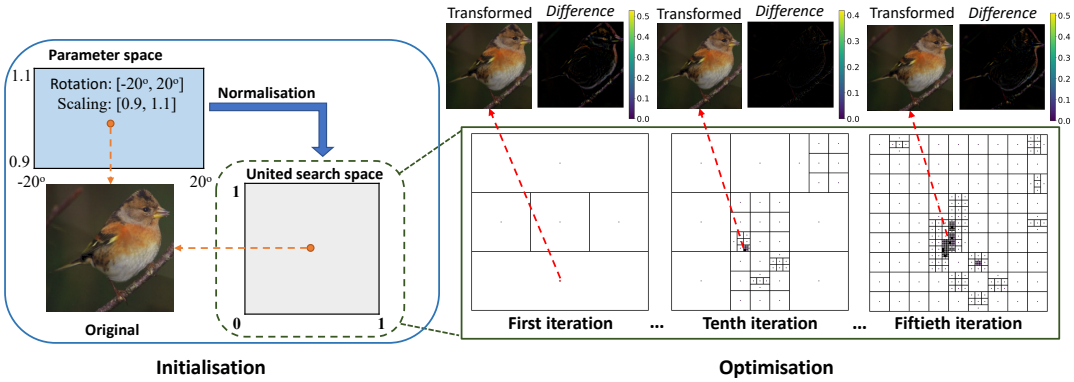


Figure 1: Schematic illustration of GeoRobust framework. After normalising the parameter space to a unit search space, GeoRobust performs a sequence of space divisions to find the global worst-case transformation.

metric transformations and an input example, GeoRobust converges to the worst-case combination for minimising an adversarial loss within a finite number of queries. We enable GeoRobust to better utilise GPUs via 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 a Lipschitz-continuous neural network, we can apply Lipschitzian optimisation to analyse their geometric robustness with guaranteed convergence.
2. We develop GeoRobust, a *query*-based 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). 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. 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 will introduce 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 design a controllable strategy 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 $\ell_{\min}$.

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 of ℓ.

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. A visualisation

[1]All appendixes of this paper can be found at https://github.com/TrustAI/GeoRobust/blob/main/appendix.pdf

of this space division procedure is presented 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 we will describe in the next section, all subspaces will get subdivided by GeoRobust after several 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 during identification.

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 at 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 existing 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) 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 analytical 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

$$\hat{K}_{c_i^+} = \frac{\|\ell(c) - \ell(c_i^+)\|}{d_i^c} \quad \text{and} \quad \hat{K}_{c_i^-} = \frac{\|\ell(c) - \ell(c_i^-)\|}{d_i^c}, \tag{14}$$

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 pretrained MNIST and CIFAR10 models released by Li et al. (2021) and utilised TIMM (Wightman 2019), 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 querying the target models' output, it can be easily deployed on any pretrained neural networks. 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 verifying on MNIST and CIFAR10, where, 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, but 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, where we report the verified accuracy determined by each baseline method. GeoRobust outperforms previous methods under all scenarios and reports the

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).

Dateset	Geo. Trans.	GeoRobust	Gsmooth	TSS	DeepG	Interval	Semantify-NN	DistSPT	TSS attack	Grid search
MNIST	$R(50^\circ)$	**98.2%**	95.7%	97.4%	$\leq$ 85.8% ($R(30^\circ)$)	$\leq$6.0% ($R(30^\circ)$)	$\leq$ 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	$R(10^\circ)$	**74.8%**	65.6%	70.6%	62.5%	20.2%	-	37%	76.4%	74.8%
	$R(30^\circ)$	**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 2: Verifying geometric robustness on ImageNet. The target model is ResNet50, which vanilla accuracy is 74%. Geometric transformations are $R(20^\circ)$, $S(0.1)$, and $T(22.4, 22.4)$.

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%

same verified accuracy as grid search. Such a performance demonstrates the effectiveness of GeoRobust in verifying the geometric robustness against 1-dimensional transformation. The evaluation on ImageNet is summarised in Tab. 2, where we can see that the accuracy verified by GeoRobust is comparable to or better than the grid search. At the same time, random pick with sufficient queries can achieve a similar performance as grid search on locating geometric adversarial examples. 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 on verifying only 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

In this section, we investigate the robustness of some popular DNN classifiers against geometric transformations. The previous section demonstrates that GeoRobust can efficiently find a worst-case combination of transformations in the black-box manner. We utilise such an advantage to test large-scale ImageNet classifiers regarding their geometric robustness against the combination of rotation $R(20^\circ)$, translation $T(22.4, 22.4)$, and scaling $S(0.1)$.

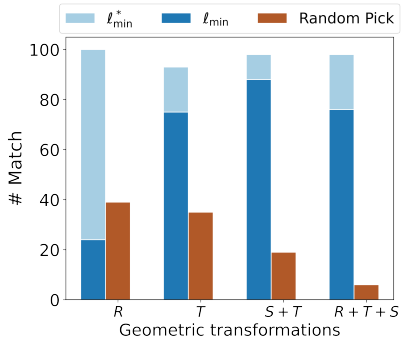
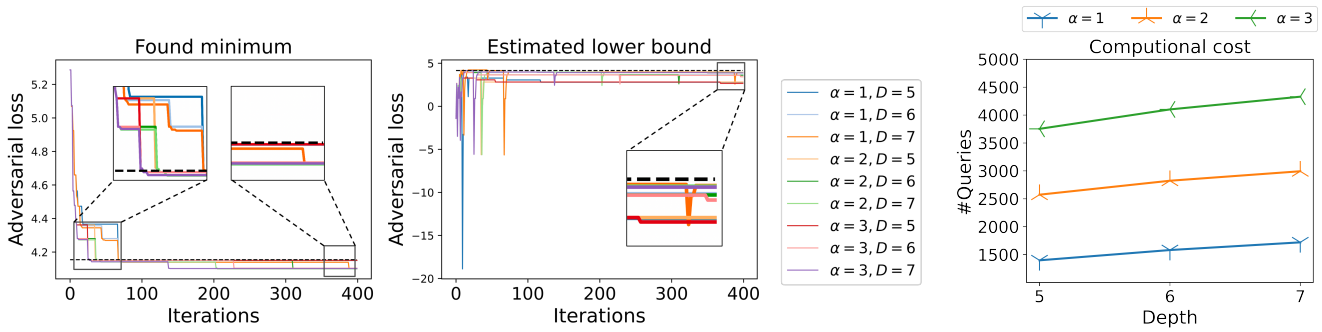


Figure 2: Comparing the global minimum found by grid search, random pick, and GeoRobust. The geometric transformations are the same as in Tab. 2. We mark an example as a match if its corresponding minimum found by a method is equal to or smaller than the minimum found by grid search.

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

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
Wide ResNet50.	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
$Vit_{32\times32}$	75.60%	23.40%	19.00%	8.8×10^7
$Vit_{16\times16}$	81.40%	41.20%	34.20%	8.6×10^7
Large $Vit_{16\times16}$	83.40%	49.20%	40.20%	3.0×10^8
$Beit_{16\times16}$.	83.80%	56.40%	52.00%	6.5×10^7
Large $Beit_{16\times16}$	**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



(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.

The results are summarised in Tab. 3, and 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, Semantify-NN, and Interval extend verification techniques designed for L_p-norm based additive perturbation. Both Semantify-NN and GSmooth 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 and TSS are also random 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 are continuous, the image pixels' coordinates are bounded integers, which means the possible outcomes for a particular set of transformations is 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 method. 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 collaboration with probabilistic approaches (Zhang, Ruan, and Fieldsend 2022) will be explored in our future works.

Conclusion

In this paper, we propose a black-box robustness analyser, GeoRobust, to efficiently verify the robustness of large-scale DNNs against geometric transformation. Given the bound of multiple geometric transformations and an input example, GeoRobust is guaranteed to 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 the proposed method. On the methodology side, we generalise the sampling strategy from DIRECT to better leverage GPU parallel computation and design an estimation method to enable GeoRobust an anytime verification. 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.

Acknowledgement.

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 Faculty of Environment, Science and Economy at University of Exeter.

We would like to thank Haozhe Wang, Anjan Dutta, and the anonymous reviewers for their helpful comments, as well as Yinlin Li for sharing us with the pre-trained models.

References

Alaifari, R.; Alberti, G. S.; and Gauksson, T. 2019. ADef: an Iterative Algorithm to Construct Adversarial Deformations. In *ICLR*.

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*.

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

Madry, A.; Makelov, A.; Schmidt, L.; et al. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*.

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 *The 32nd British Machine Vision Conference (BMVC'21)*.

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 *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, 2651–2659.

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 *The 28th International Joint Conference on Artificial Intelligence (IJCAI'19)*. IJCAI.

Ruan, W.; Yi, X.; and Huang, X. 2021. Adversarial Robustness of Deep Learning: Theory, Algorithms, and Applications. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM'21)*, 4866–4869.

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*.

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. In *HANDBOOK ON COMPUTER LEARNING AND INTELLIGENCE: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation*, 547–584.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018. Formal security analysis of neural networks using symbolic intervals. In *USENIX Security*.

Wang, Z.; and Ruan, W. 2022. Understanding Adversarial Robustness of Vision Transformers via Cauchy Problem. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'22)*.

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*.

Wightman, R. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models.

Xiang, W.; Tran, H.; and Johnson, T. T. 2017. Reachable set computation and safety verification for neural networks with relu activations. *arXiv*, abs/1712.08163.

Xiao, C.; Zhu, J.; Li, B.; et al. 2018. Spatially Transformed Adversarial Examples. In *ICLR*.

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.; and Xu, P. 2023. Reachability Analysis of Neural Network Control Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'23)*.

Zhang, T.; Ruan, W.; and Fieldsend, J. E. 2022. PRoA: A Probabilistic Robustness Assessment against Functional Perturbations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'22)*.

Zhang, Y.; Ruan, W.; Wang, F.; and Huang, X. 2020. Generalizing universal adversarial attacks beyond additive perturbations. In *2020 IEEE International Conference on Data Mining (ICDM'20)*, 1412–1417. IEEE.

Appendix

Algorithm pseudocode

Algorithm 1: GeoRobust

Input: An input example x, the objective function ℓ, the bound of the parameter space P, the number of function evaluation Q, the number of iterations T, the maximum depth D, the size of candidates set α

Output: $\ell_{\min}$ with the corresponding solution $c_{\min}$ and an estimation of the ground truth minimum $\ell^*_{\min}$

Normalise the parameter space to a unit hypercube with centre point c_0
$t \leftarrow 0$, $q \leftarrow 0$
Initialise the index set of hyperrectangles $\mathcal{H} = \{0\}$
Initialise the set of potential optimal space $\mathcal{P} = \{0\}$
while $(t \leq T) \cap (q < Q) \cap (\mathcal{P} \neq \varnothing)$ **do**
    Initialise $\mathcal{X} = \{\}$
    for *each potential optimal hyperrectangle p in $\mathcal{P}$* **do**
        if *hyperrectangle size* $\sigma_p = 3^{-D}$ **then**
            Continue
        else
            for *each dimension i with long edge of hyperrectangle p* **do**
                Append$(\mathcal{X}, c_p \pm \delta_j^{c_p} \mathbf{e}_i)$
                Append$(\mathcal{H}, \{q+1, q+2\})$
                q += 2
    /* Conduct function evaluation via a single forward propagation */
    $\mathcal{Y} = \ell(\mathcal{X})$
    /* Space division */
    for *each potential optimal hyperrectangle p in $\mathcal{P}$* **do**
        Subdivide hyperrectangle p based on query results in $\mathcal{Y}$
        Recording the size σ and local slope $\hat{K}$ for all new generated subspaces
        Update p's size σ_p and local slope $\hat{K}_{c_p}$
    /* Record current best evaluation and corresponding solution */
    $\ell_{\min} = \min_{q \in \mathcal{H}} \ell(c_q)$, and $c_{\min} = \arg\min_{c_q} \ell(c_q)$
    Estimate the ground truth $\ell^*_{\min}$ via Eq. (15)
    /* Select potential optimal subspaces */
    Reset $\mathcal{P} = \{\}$
    for $d \in \{0, 1, \ldots, D-1\}$ **do**
        Build candidates set $\mathcal{H}_\alpha$ from hyperrectangles with $\sigma = 1/3^d$
        for *each hyperrectangle q in $\mathcal{H}_\alpha$* **do**
            if *q satisfies condition* (11) **then**
                Append$(\mathcal{P}, q)$
    t = t+1

Proofs

Proof of Lemma 1

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.

Proof. The derivative of pixel value V_i w.r.t. $\mathtt{x}_i$ is given by

$$\frac{\partial V_i}{\partial \mathtt{x}_i} = \sum_n^H \sum_m^W U_{nm} \max(0, 1 - |\mathtt{y}_i - n|) \times \begin{cases} 0 & \text{if } |m - \mathtt{x}_i| \geq 1, \\ 1 & \text{if } |m - \mathtt{x}_i| < 1 \text{ and } m \geq \mathtt{x}_i, \\ -1 & \text{if } |m - \mathtt{x}_i| < 1 \text{ and } m < \mathtt{x}_i. \end{cases} \tag{16}$$

There are only four neighbouring pixels satisfy $|m - \mathtt{x}_i| < 1$ and $|\mathtt{y}_i - n| < 1$, so Eq. (16) can then be written as

$$\begin{aligned} \frac{\partial V_i}{\partial \mathtt{x}_i} &= U_{\underline{n}\bar{m}} \cdot (1 - y_i + \underline{n}) + U_{\bar{n}\bar{m}} \cdot (1 - \bar{n} + y_i) \\ &\quad - U_{\bar{n}\underline{m}} \cdot (1 - \bar{n} + y_i) - U_{\underline{n}\underline{m}} \cdot (1 - y_i + \underline{n}) \\ &= (1 - y_i + \underline{n}) \left(U_{\underline{n}\bar{m}} - U_{\underline{n}\underline{m}}\right) \\ &\quad + (1 + y_i - \bar{n}) \left(U_{\bar{n}\bar{m}} - U_{\bar{n}\underline{m}}\right), \end{aligned} \tag{17}$$

where $(\bar{n}, \bar{m}) = (\lceil y_i \rceil, \lceil x_i \rceil)$ and $(\underline{n}, \underline{m}) = (\lfloor y_i \rfloor, \lfloor x_i \rfloor)$. We can see that

$$(1 - y_i + \underline{n}) + (1 + y_i - \bar{n}) = 1, \tag{18}$$

which means Eq. (17) is taking a weighted average of the difference between two pairs of pixels. Without loss of generality, suppose the eligible pixel value is in $[0, 1]$. We have

$$\sup_{\mathtt{x} \in x} \left(\frac{\partial V_i}{\partial \mathtt{x}}\right) = 1, \tag{19}$$

and following a similar deduction, the same result can be obtained for $\frac{\partial V_i}{\partial \mathtt{y}}$. Then, the derivatives of $\mathtt{x}$ and $\mathtt{y}$ w.r.t. transformation matrix A_θ are

$$\frac{\partial \mathtt{x}_i}{\partial A_\theta} = \begin{bmatrix} \frac{\partial \mathtt{x}_i}{\partial \theta_{11}} & \frac{\partial \mathtt{x}_i}{\partial \theta_{12}} & \frac{\partial \mathtt{x}_i}{\partial \theta_{13}} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathtt{x}'_i & \mathtt{y}'_i & 1 \\ 0 & 0 & 0 \end{bmatrix}, \tag{20}$$

and

$$\frac{\partial \mathtt{y}_i}{\partial A_\theta} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{\partial \mathtt{y}_i}{\partial \theta_{21}} & \frac{\partial \mathtt{y}_i}{\partial \theta_{22}} & \frac{\partial \mathtt{y}_i}{\partial \theta_{23}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ \mathtt{x}'_i & \mathtt{y}'_i & 1 \end{bmatrix}. \tag{21}$$

For each θ, we have

$$\sup_{\theta \in A_\theta} \left(\frac{\partial \mathtt{x}_i}{\partial \theta}\right) = W, \text{ and } \sup_{\theta \in A_\theta} \left(\frac{\partial \mathtt{y}_i}{\partial \theta}\right) = H. \tag{22}$$

In the final step, let us take the scaling factor λ as an example. Following the chain rule, the partial derivative is given by

$$\frac{\partial V_i}{\partial \lambda} = \frac{\partial V_i}{\partial x_i} \frac{\partial x_i}{\partial \theta_{11}} \frac{\partial \theta_{11}}{\partial \lambda} + \frac{\partial V_i}{\partial y_i} \frac{\partial y_i}{\partial \theta_{22}} \frac{\partial \theta_{22}}{\partial \lambda}. \tag{23}$$

Let $\mathcal{R}$ be the set of all eligible γ, we can substitute Eq. (19) and (22) into Eq. (23) and bound the derivative as

$$\frac{\partial V_i}{\partial \lambda} \leq \sup_{\gamma \in \mathcal{R}} (\cos \gamma) \cdot (W + H). \tag{24}$$

Because there are finite numbers of pixels, the overall derivative has an upper bound as well. Similarly, by specifying the range of each transformation factor, their derivatives are upper bound correspondingly, and this completes the proof. □

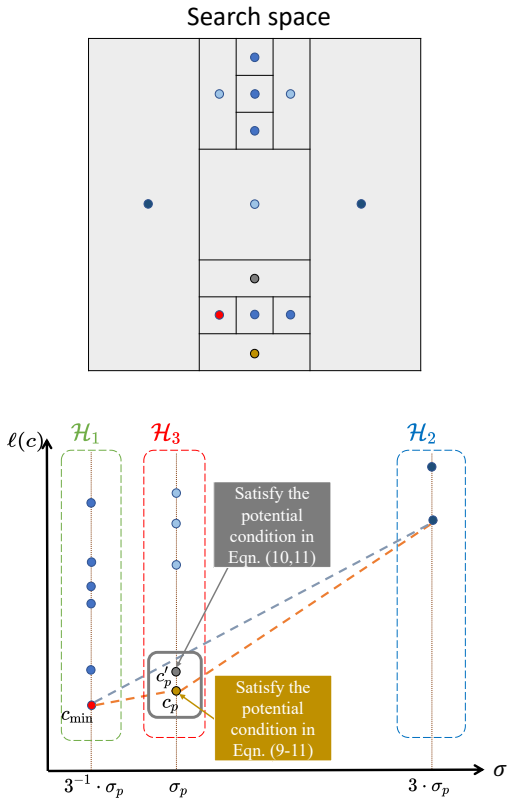


Figure 4: A visualisation about potential optimal condition (7) and α candidate set ($\alpha = 2$). A partition of the search space is presented in the upper figure, and the relationship between the sizes and corresponding function values of all subspaces is plotted in the lower figure. GeoRobust would select both c_p and c_p' as PO subspaces.

Proof of Lemma 2

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

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

Proof. For a hyperrectangle p, we can group all hyperrectangles into $\mathcal{H}_1^p$, $\mathcal{H}_2^p$, and $\mathcal{H}_3^p$, then inequation (7) can be rewritten into three inequalities,

$$\tilde{K} \geq \frac{\ell(c_j) - \ell(c_i)}{\sigma_j - \sigma_i}, \forall i \in \mathcal{H}_1^j, \tag{25}$$

$$\tilde{K} \leq \frac{\ell(c_i) - \ell(c_j)}{\sigma_i - \sigma_j}, \forall i \in \mathcal{H}_2^j, \tag{26}$$

and inequation (9). Putting inequalities (25) and (26) together gives inequity (10). If a hyperrectangle satisfies inequalities (9) and (10) at the same time, then the PO condition (7) is satisfied. While we do not know the true $\tilde{K}$ in Eq. (8), it can be replaced by an upper bound given in (26). Substituting condition (26) into condition (8) gives us inequalities (11). This completes the proof. □

Explanation of Definition 1

We encourage GeoRobust to select more PO subspace via remove inequation (9). A visualisation of α candidate set is presented in Fig. 4, where both c_p and c_p' would be selected and queried by GeoRobust, while DIRECT optimisation would only choose c_p.

Explanation of Remark 1

In every iteration, there is a hyperrectangle p satisfies $\sigma_p = \max_{q\in\mathcal{H}} \sigma_q$ and $\ell(c_p) = \min_{q\in\mathcal{H}_3^p} \ell(c_q)$. Please recall that $\mathcal{H}_3^p$ contains the indexes of hyperrectangles with the same size as p and $\mathcal{H}_1^p$ contains the indexes of hyperrectangles that are smaller than p, while $\mathcal{H}_2^p$ is empty because no hyperrectangle larger than p exists. Because $\mathcal{H}_2^p = \varnothing$, PO condition (10) only produces a lower bound on $\tilde{K}$, *i.e.*,

$$\max_{i\in\mathcal{H}_1^j} \frac{\ell(c_j) - \ell(c_i)}{\sigma_j - \sigma_i} \leq \tilde{K}, \tag{27}$$

which means we can always find a slope that is large enough to satisfy PO conditions, making hyperrectangle p a PO subspace. Therefore, GeoRobust would identify and partition at least one PO hyperrectangle throughout each iteration. Furthermore, for any hyperrectangle q, there is only a finite number of hyperrectangles in its $\mathcal{H}_2^q$ and $\mathcal{H}_3^q$. Under the worst situation, hyperrectangle q will be selected as a PO space and get divided in the next iteration when $\mathcal{H}_2^q \cup \mathcal{H}_3^i = \{q\}$.

Proof of Theorem 1

To prove Theorem 1, we need a relationship between the depth of the largest subspace and the number of queries, which is given in the Theorem 4.2 from (Gablonsky 2001).

Theorem 4.2. (Gablonsky 2001) Assuming that only one hyperrectangle gets divided in every iteration, the number of iterations T after which no hyperrectangle of depth $d-1$ is left is given by

$$T = 3^{n-1}\left(\frac{3^{nd}-1}{3^n-1}\right) < 3^{nd} - 1. \tag{28}$$

We can now prove 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 Q queries 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, the analytical complexity is given by $\mathcal{O}\big((\tilde{K}/\varepsilon)^n\big)$.

Proof. Due to the Lipschitz continuity, we have

$$\ell_{\min} - \min_{c \in C} \ell(c) \leq \epsilon \leq \tilde{K} \cdot 3^{-d}, \tag{29}$$

where d is the depth of the current largest subspace in the unit search space. According to Eq. (28), we have $d \geq \frac{log_3(T+1)}{n}$, and substituting it into Eq. (29) gives

$$\varepsilon \leq \tilde{K} \cdot 3^{-\frac{log_3(T+1)}{n}} = \tilde{K} \cdot (T+1)^{-\frac{1}{n}}. \tag{30}$$

Following the same assumption in Theorem 4.2 (Gablonsky 2001), we suppose that only one hyperrectangle gets divided in every iteration. Then we have

$$\varepsilon \leq \tilde{K} \cdot (T+1)^{-\frac{1}{n}} < \tilde{K} \cdot (T+1)^{-\frac{1}{n}}, \tag{31}$$

which leads to Eq. (14). The relationship between any desired ε and the number of iterations T is then given by

$$n \cdot (\tilde{K}/\varepsilon)^n - n > Q, \tag{32}$$

which shows the analytical complexity is $\mathcal{O}\big((\tilde{K}/\varepsilon)^n\big)$. This completes the proof. □

Detailed related works

Numerous studies have been conducted to find the worst-case adversarial perturbation. While several adversarial attacks, such as the projected gradient descent attack (Madry et al. 2018), and Auto Attack (Croce and Hein 2020), can generate strong adversarial examples, they cannot ensure finding the worst-case perturbation (Huang et al. 2020). Some complete verification technologies can be used to find the worst-case perturbation (Liu et al. 2021), where completeness means that a method is guaranteed to find adversarial examples within a given norm ball unless no adversarial example exists, but most of them are computationally inefficient and have specific requirements for their target models. ExactReach (Xiang, Tran, and Johnson 2017) and ReluVal (Wang et al. 2018), for example, perform layer-by-layer propagation through target models with only linear or ReLU activations, requiring their target models to be fully accessible. Therefore, these methods only work under the white-box setting and are unsuitable for large-scale neural networks. Apart from the limitation on scalability, the layer-by-layer propagation operation needs a L_p norm based pixel-level or element-level bounding box of the input. As illustrated in Fig. 1, it is difficult to establish such a bounding box for geometric transformations because even a small transformation could affect a huge number of pixels and drastically alter their value. DeepGo (Ruan, Huang, and Kwiatkowska 2018) is a global optimisation based method that operates under the grey-box environment, *i.e.*, requiring no knowledge of the model's parameters but a pre-estimation of the model's Lipschitz constant, which is difficult to get in reality. Due to space limitations, we cannot cover all complete verification methods here and refer readers to a recent survey on verification techniques (Liu et al. 2021).

On the other hand, there are also some studies on the geometric robustness of DNNs, and we summarise the difference between our method and related works in Table 4 on finding the worst-case geometric transformation. Jaderberg et al. (2015) proposed a differential module called spatial transformer network (STN) to enhance neural networks' learning ability regarding geometric transformations. Although the parameter space of control factors for most geometric manipulations is continued, the image pixels' coordinates are discrete, bounded integers, which means the possible outcomes for a particular set of transformations are finite. Pei *et al.* (Pei et al. 2017) empirically evaluated DNNs' resistance toward geometric transformations by enumerating all possible values. Similarly, Engstrom *et al.* (Engstrom et al. 2019) employed random pick and grid search to discover the adversarial translation and rotation to deceive target models. DeepG (Balunović et al. 2019) computes a convex relaxation of the bounding box for a set of geometric transformations and then certifies the robustness property via existing robustness verifier (Singh et al. 2019). Mophapatra *et al.* (Mohapatra et al. 2020) introduced a small network, called Semantify-NN, to simulate the geometric manipulation and adopted existing verifier (Weng et al. 2018) to examine the hybrid model composed of Semantify-NN and a target network. Because FastLin (Weng et al. 2018) and DeepPoly (Singh et al. 2019) are incompleteness verifiers, these two works may certify whether a set of geometric transformations can affect the predictions of a target classifier but are unable to determine the worst-case transformations precisely. Besides, these two verifiers use layer-by-layer propagation, which is computationally inefficient and limited to small networks in the white-box setting. DistSPT (Fischer, Baader, and Vechev 2020), TSS (Li et al. 2021), and GSmooth (Hao et al. 2022) utilise random smoothing techniques to verify the geometric robustness. TSS is a black-box verification method that is based on random smoothing. DistSPT combines random smoothing and interval bound propagation together to conduct the verification on tasks beyond L_p norm. GSmooth also uses an image-to-image network to simulate the geometric transformation.

Parallel to geometric transformation, several works (Alaifari, Alberti, and Gauksson 2019; Xiao et al. 2018) investigated spatial transformation, which is a spatial distortion of the coordinates of pixels. Please note that spatial transformations performed using vector fields remain pixel-level perturbation. Thus, it is fundamentally distinct from geometric transformation and beyond the scope of this paper.

Experiments

Implementation details within Tab. 1

To make the comparison, we use GeoRobust to verify the same models used in (Li et al. 2021) on the same subsets of MNIST, CIFAR-10, and ImageNet. In Tab. 5, we present the benign accuracy reported by (Li et al. 2021) and obtained on our machine. It can be seen that the reproduced accuracy of MNIST and CIFAR-10 models are basically consistent with the reported accuracy, while the ImageNet models are much less accurate than expected (corresponding code is provided for reviewing). Since we failed to load the ImageNet models properly, the comparison was only done on MNIST and CIFAR-10 datasets in Tab. 1.

Table 4: Comparison of methods for finding the worst-case transformation and providing the lower bound for verification.

Method	Approach	Requirement	Efficiency	Scalability		Guarantee	
				Architecture	Scale	Lower bound	Worst-case
Exhaustive search (Pei et al. 2017)	Query	None	✗	✓	✓	✗	✓
Random pick (Engstrom et al. 2019)	Query	None	✓	✓	✓	✗	✗
DeepG (Balunović et al. 2019)	Layer-by-layer propagation	Specify transformation access all parameters	✗	✗	✗	✓	✗
Semantify-NN (Mohapatra et al. 2020)	Surrogate network and layer-by-layer propagation	Specify transformation access all parameters	✗	✗	✗	✓	✗
DistSPT (Fischer, Baader, and Vechev 2020)	Random smoothing and Layer-by-layer propagation	Specify transformation access all parameters	✗	✓	✓	✓	✗
TSS (Li et al. 2021)	Random smoothing	Specify transformation	✗	✓	✓	✓	✗
GSmooth (Hao et al. 2022)	Surrogate network and random smoothing	Specify transformation access all parameters	✗	✓	✗	✓	✗
GeoRobust (ours)	Query	None	✓	✓	✓	✓	✓

Table 5: Benign accuracy of models trained and verified by Li et al. (2021). The small CNN used for MNIST classification has 4 convolutional layers and 3 fully connected layers.

Model	Dataset	Transf.	Reported acc.	Reproduced acc.
small CNN	MNIST	$R(50^\circ)$	99.4%	99.4%
		$S(0.3)$	99.4%	99.4%
ResNet101	CIFAR-10	$R(10^\circ)$	83.2%	84%
		$R(30^\circ)$	82.6%	81.2%
		$S(0.3)$	79.8%	80.8%
ResNet50	ImageNet	$R(30^\circ)$	46.2%	20.8%
		$S(0.3)$	50.8%	26.6%

GeoRobust is carried out with $D = 5$ and $\alpha = 2$, and its computational budget is up to 200 iterations and 2000 queries. In practice, GeoRobust only conducted 244 queries on average to verify the 1-dimensional adversarial geometric transformations. The average runtime on MNIST and CIFAR-10 are 0.18 and 0.74 seconds, respectively. The grid search, here, is carried out with 2000 queries, which is sufficient for exploring 1-dimension transformation.

Additional experiments on all combinations of geometric transformation In Tab. 2, we compared GeoRobust to random pick and grid search under four combinations of geometric transformations, while the comparison on all combinations of geometric transformations is summarised in Tab. 6. GeoRobust is carried out with $D = 6$ and $\alpha = 2$, and the computational budget is up to 150 iterations and 2000 queries. It can be seen from Tab. 6 that GeoRobust is significantly more efficient than random pick and grid search under all settings. While three methods perform similarly on verifying 1-dimensional geometric transformations, GeoRobust can achieve the same and sometimes even better performance than grid search when verifying multiple transformations together.

Detailed benchmark on ImageNet classifiers We present a completed version of Tab. 3, presenting the average numbers of queries and runtime. Here GeoRobust can run up to 150 iterations and 3000 queries per example. The depth and candidate set are set to be $D = 6$ and $\alpha = 2$. The geometric transformations are $R(20^\circ)$, $S(0.1)$, and $T(22.4, 22.4)$. In Tab. 7, we can see that GeoRobust can conduct super efficient analysis on all ImageNet classifiers, and it only takes GeoRobust less than 11 seconds to analyse one example on the large Vit with 3×10^8 parameters, which is the largest model here.

Table 6: Verifying geometric robustness on ImageNet against all combination of three transformations:$R(20^\circ)$, $S(0.1)$, and $T(22.4, 22.4)$. The target model is ResNet50, which achieves 74% classification accuracy. To make a fair comparison on efficiency, the random pick and grid search are also implemented on GPU, where the batch size is 128.

Transformations	GeoRobust			Random pick			Grid search		
	Verified Acc.	#Queries	Runtime (s)	Verified Acc.	#Queries	Runtime (s)	Verified Acc.	#Queries	Runtime (s)
R	58%	667 ± 29	2.4	58%	2000	4.8	58%	2000	4.7
S	59%	679 ± 30	2.4	59%		4.7	59%		4.7
$R+S$	**54%**	1096 ± 151	3.8	56%	4000	9.6	56%	5000	12.1
T	57%	1046 ± 99	3.6	57%		9.4	59%		11.9
$R+T$	**46%**	1187 ± 112	4.1	51%	6000	14.4	49%	1×10^4	29.3
$S+T$	**57%**	1170 ± 111	4.0	60%		14.1	57%		28.6
$R+S+T$	**46%**	1295 ± 117	4.5	49%	8000	19.1	46%	1×10^5	251.3

Table 7: A completed version of Tab. 3: Benchmarking Geometric Robustness on ImageNet

Models	Clean	Attack	Verified	#Parameters	# Queries	Runtime (s)
Inception v3.	73.60%	28.20%	24.20%	2.4×10^7	1405±205	4.6±0.5
Inception v3$_{adv}$	75.00%	30.60%	27.00%	2.4×10^7	1398±192	4.6±0.5
Inception v4	78.40%	40.20%	36.40%	4.3×10^7	1444±256	6.3±0.9
ResNet34	64.40%	10.60%	9.00%	2.2×10^7	1475±245	3.1±0.4
ResNet50	78.40%	54.00%	31.12%	2.6×10^7	805±110	4.8±0.6
Wide ResNet50.	81.60%	49.40%	40.00%	6.9×10^7	1283±125	6.0±0.5
ResNet101	80.00%	54.20%	48.20%	4.5×10^7	1291±134	5.8±0.5
ResNet152	79.40%	53.80%	46.20%	6.0×10^7	1278±129	7.3±0.6
Vit$_{32\times32}$	75.60%	23.40%	19.00%	8.8×10^7	1528±297	3.3±0.5
Vit$_{16\times16}$	81.40%	41.20%	34.20%	8.6×10^7	1471±268	5.0±0.8
Large Vit$_{16\times16}$	83.40%	49.20%	40.20%	3.0×10^8	1410±244	10.4±1.6
Beit$_{16\times16}$.	83.80%	56.40%	52.00%	6.5×10^7	1403±215	4.9±0.7
Large Beit$_{16\times16}$	**85.60%**	**65.60%**	**58.20%**	2.3×10^8	1363±190	10.5±1.3
Gmlp	77.96%	40.80%	36.80%	1.9×10^7	1661±417	4.3±0.9
Mixer	72.20%	27.20%	23.40%	6.0×10^7	1566±337	4.5±0.9
Swin	80.20%	34.60%	13.20%	8.8×10^7	1292±136	5.4±0.5
Xcit	76.80%	40.40%	20.60%	8.4×10^7	1497±355	6.5±1.1
Pit	79.40%	36.60%	20.00%	7.4×10^7	1538±371	11.0±1.1