Complex & Intelligent Systems (2023) 9:3801–3818
https://doi.org/10.1007/s40747-022-00790-x

ORIGINAL ARTICLE

Quantifying safety risks of deep neural networks

Peipei Xu[1] · **Wenjie Ruan**[2] · **Xiaowei Huang**[1]

Received: 16 June 2021 / Accepted: 1 May 2022 / Published online: 9 July 2022
© The Author(s) 2022

Abstract
Safety concerns on the deep neural networks (DNNs) have been raised when they are applied to critical sectors. In this paper, we define safety risks by requesting the alignment of network's decision with human perception. To enable a general methodology for quantifying safety risks, we define a generic safety property and instantiate it to express various safety risks. For the quantification of risks, we take the maximum radius of safe norm balls, in which no safety risk exists. The computation of the maximum safe radius is reduced to the computation of their respective Lipschitz metrics—the quantities to be computed. In addition to the known adversarial example, reachability example, and invariant example, in this paper, we identify a new class of risk—uncertainty example—on which humans can tell easily, but the network is unsure. We develop an algorithm, inspired by derivative-free optimization techniques and accelerated by tensor-based parallelization on GPUs, to support an efficient computation of the metrics. We perform evaluations on several benchmark neural networks, including ACSC-Xu, MNIST, CIFAR-10, and ImageNet networks. The experiments show that our method can achieve competitive performance on safety quantification in terms of the tightness and the efficiency of computation. Importantly, as a generic approach, our method can work with a broad class of safety risks and without restrictions on the structure of neural networks.

Keywords Adversarial examples · Lipschitz metrics · Neural networks · Robustness · Safety · Uncertainty

Arabic Keywords الشبكات · العصبية · العميقة · المتانة

Introduction

In recent years, we witness significant progress that has been made in AI, especially the deep neural networks that can achieve surprisingly high performance on various tasks, including image recognition [44], natural language processing [26], and games [45]. As a key component, deep neural networks (DNNs) have also been widely used in a range of safety-critical applications such as fully or semi-autonomous vehicles [31,57], drug discovery [53], and automated medical diagnosis [12]. The applications of neural networks in safety-critical systems bring a new challenge. As recent research demonstrated [16,35,49,63], despite achieving high accuracy, DNNs are vulnerable to adversarial examples, i.e., adding a small perturbation to a genuine image will result in an erroneous output. Such phenomena essentially imply that neural network's accuracy and its robustness may not be positively correlated [52]. As a result, it is extremely crucial that a neural network model can be practically evaluated on its safety and robustness [21,43].

Many research efforts have been directed towards developing approaches to evaluate neural network's robustness by crafting adversarial examples, [2,13,23,34,64], including notably FGSM [49], JSMA [36], C&W [10], etc. These approaches can only falsify robustness claims, yet cannot verify, because no theoretical guarantee is provided on their results. Originated from verification community recently, some research works have instead focused on robustness evaluation with rigorous guarantees [17,37]. These techniques rely on either a reduction to a constraint solving problem by encoding the network as a set of constraints [25,29], an exhaustive search of the neighborhood of an image [22,58], a reduction to a global optimisation problem [41,42,55], or

✉ Wenjie Ruan
w.ruan@exeter.ac.uk

Peipei Xu
peipei.xu@liverpool.ac.uk

Xiaowei Huang
xiaowei.huang@liverpool.ac.uk

[1] Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK

[2] Department of Computer Science, University of Exeter, Exeter EX4 4QF, UK

an over-approximation method [14], etc. However, these approaches can only work with small-scale neural networks in a white-box manner,[1] and have not been able to work with state-of-the-art neural networks such as various ImageNet models. Moreover, recent works explore efficient verifiers by layer-by-layer convex relaxation [8,61]. However, existing approaches may lack of generality to quantify different safety risks and can only work on a particular single safety risk, such as local or point-wise robustness. For more details on safety properties, please refer to our recent survey [20,21].

In this regard, this paper works towards a generic quantification framework that is able to (i) work with different classes of safety risks, such as robustness, reachability and uncertainty; (ii) provide guarantee on its quantification results; and (iii) applicable to large-scale neural networks with a broad range of layers and general activation functions. To achieve these goals, we introduce a generic property expression parameterised over the output of a DNN, define metrics over this expression, and develop a tool DEEPQUANT to evaluate the metrics on DNNs. By instantiating the property expression with various specific forms and consider different metrics, DEEPQUANT can evaluate different safety risks on neural networks including the local and global robustness, as well as the decision uncertainty, a new type of safety risks that is first studied in this paper. Specifically, the key technical contributions of this paper lie on the following aspects.

First, we study safety risks by assuming that network's decision needs to align with human perception in "Safety risks in neural networks". Under this assumption, we identify another class of safety risks other than the known ones—adversarial example [49], reachability example [11,41], and invariant example [24]—and name it as uncertainty example. Figure 1 presents the intuition of these safety risks. Different from adversarial example on which the network is certain about its decision (although the decision is incorrect w.r.t. human perception), uncertainty example lies on the vicinity of the intersection point of all decision boundaries (marked by red dashed line circle in Fig. 1a) and should be without any confusion with human perception. Uncertainty is more difficult to evaluate than robustness, because the intersection areas of *all* decision boundaries are very sparse in the input space. Figure 1b shows the output results of uncertainty examples compared with adversarial examples on MNIST dataset. Moreover, the potential disastrous consequence of uncertainty example will be discussed in "Safety risks in neural networks".

Second, to work with different safety risks in a single framework, we define in "Quantification of safety risks" a generic safety property expression and show that it can be instantiated to express various risks. The quantification of the risks is then defined as the maximum radius of safe norm balls, in which no risk is present. Then, we show that a *conservative* estimation of the maximum radius can be done by computing a Lipschitz metric over the safety property.

Third, we develop in "Safety risk quantification algorithms" an algorithm, inspired by a derivative-free optimisation technique called Mesh Adaptive Direct Search (MADS), to compute the Lipschitz metric. The algorithm is able to work on large-scale neural networks and does not require knowledge about the internal weights or structures of DNNs. Moreover, as indicated in Fig. 3 in "Safety risk quantification algorithms" , our algorithm is tensor-based, so it is able to take advantage of the significant capability of GPU parallelisation.

Finally, we implement the approach into a tool DEEPQUANT[2] and validate it over an extensive set of networks, including large-scale ImageNet DNNs with millions of neurons and tens of layers. The experiments in "Experimental results" show competitive performance of DEEPQUANT in a number of benchmark networks with respect to current verification tools such as ReluPlex [25], SHERLOCK [11], and DeepGO [41]. Our method can work without restrictions on the safety properties and the structure of neural networks. This is in contrast with existing tools; for example, ReluPlex and SHERLOCK can only work with small network with ReLU activation functions and DeepGO can only work with robustness and reachability. We also discuss our result in "Discussion". In summary, the novelty of this paper lies on the following aspects:

- This paper introduce a generic property expression that provides a principled and unified tool to quantify various safety risks on deep neural networks;
- We prove that the proposed Lipschitzian robustness expression can approximate the true robustness in terms of classification-invariant space;
- This paper, as the the first research work, identifies a new type of risk of neural networks called uncertainty examples, as well as provides an efficient method to identify such uncertainty spots;
- We implement the proposed solution as a software tool—DEEPQUANT that can quantitatively measure its robustness as well as the uncertainty, and return adversarial examples or uncertainty examples if exist, for a given neural network and a concrete L_p-norm ball. In addition, DEEPQUANT is applicable to large-scale deep neural networks including various ImageNet models.

To improve the readability of our paper, we illustrate the proposed framework in Fig. 2. Specifically, DEEPQUANT consists of three key parts, including

[1] Namely, the structure and the internal weights of DNNs need to be known.

[2] The software is provided via github: https://github.com/TrustAI/DeepQuant.

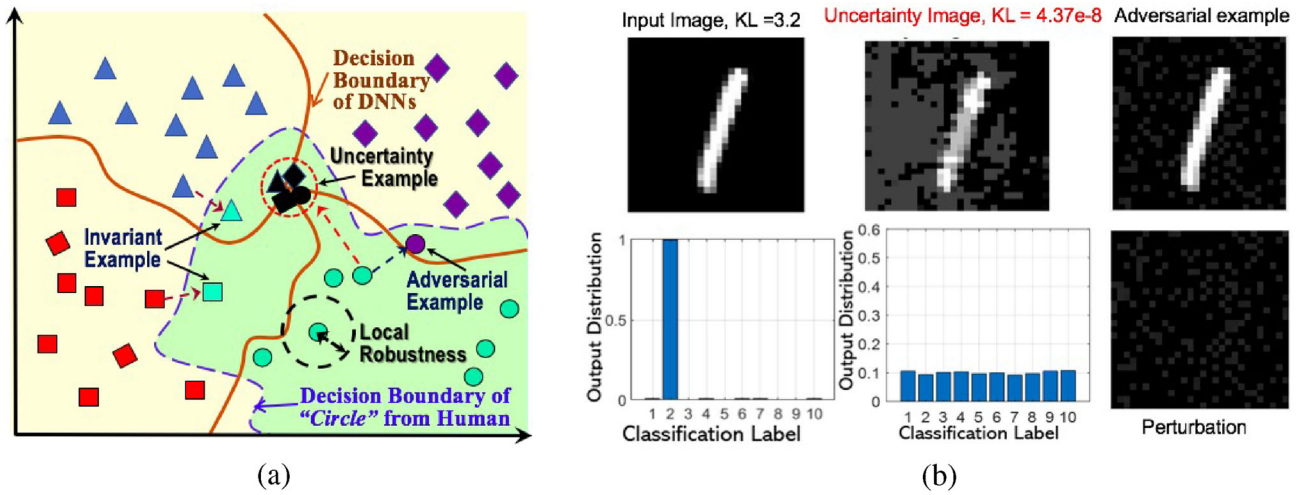


Fig. 1 **a** Illustration of three safety risks—adversarial example [49], invariant example [24], and uncertainty example (this paper [60]). **b** An example to compare uncertainty example with adversarial example in MNIST. The first row: the first image is the raw input image, the second image is the uncertainty example (identified by our tool), and the third image is the adversarial example; the second row: the corresponding output probabilistic distributions of DNNs on raw input image and uncertainty example, and the adversarial perturbation

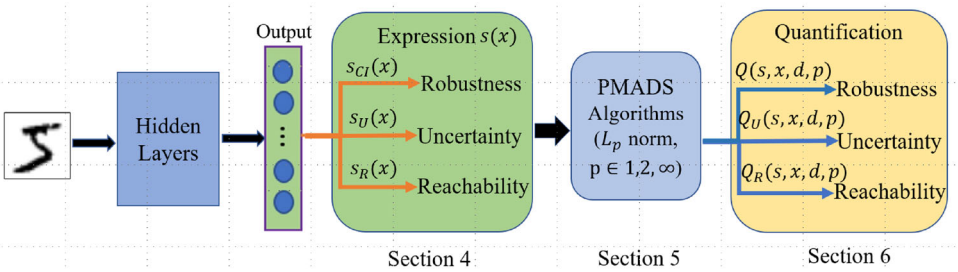


Fig. 2 The framework of DEEPQUANT (Section 4: Quantification of safety risks; Section 5: Safety risk quantification algorithms; Section 6: Experimental results)

– Instantiating safety property expression $s(x)$ including specific robustness, uncertainty, and reachability expressions in "Quantification of safety risks";
– Achieving risk quantification based on parallelized mesh adaptive direct search (PMADS) in "Safety risk quantification algorithms";
– Presenting safety quantification results including robustness, uncertainty and reachability quantification in "Experimental results".

Related work

We now discuss some of the closely related work in safety properties of neural networks.

Adversarial attacks

As recent works show that neural networks with model parameters are vulnerable to adversarial examples [49]. There are constantly increasing number of attacks to generate adversarial examples with new countermeasures [51], which become extremely awful when adversary attacks a black-box model [33]. Hence, how to improve the robustness of neural networks is a very critical task, especially for safety-critical applications. Crafting adversarial examples is a very intuitive way to evaluate model robustness, which can indicate the potential risks for neural networks. Starting from Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [49], a number of adversarial attack algorithms have been developed, including notably FGSM [16], JSMA [36], C&W attacks [10], RecurJac [62], one-pixel attacks [46], structured attack [59], binary attack [13], etc..

However, most of these works are guided by the forward gradient or the gradient of the cost-function, which in turn rely on the existence of first-order derivative, i.e., differentiability of neural network. Computing gradient is also very time-consuming. In addition, these threat models cannot work on a black-box setting when attacker is not allowed to access model parameters except for the output. To solve this issue, the method proposed in this paper relaxes this assumption and can work with any neural networks. Moreover, while adversarial attacks can falsify the robustness of a neural networks, our method can also verify the robustness, thanks to

its theoretically grounded approach of taking a Lipschitzian metric with confidence-interval expression as an indicator of the robustness. Finally, beyond robustness, our metric is generic and can express other properties, such as uncertainty and reachability, which is shown in Fig. 2.

Safety/formal verification

Another evaluating approach against adversarial attack is using verification. How to verify whether a given/particular neural network satisfies certain input–output properties is a very challenging task. Traditional verification of neural networks mainly focus on measuring the networks on a large collections of points in the input space and checking whether the outputs are as desired. However, due to the infinite of input space, it is not workable to check all possible inputs. For example, it is NP-complete complexity even solving a simple neural network only with ReLU activation functions [25]. Besides, some networks may be vulnerable to adversarial attacks, although they can perform well on a large sample of inputs and not correctly extend to new situations, which lacks a theoretical guarantee to ensure safety of systems. The recent advances of neural network are leveraging verification approaches to provide guarantees on the obtained results. The existing works include the layer-by-layer exhaustive search approach [22], methods using constraint solvers [25,39], global optimisation approaches [41,42,55,58], the abstract interpretation approach [14,28,32], linear programming (LP) [56] or mixed-integer linear programming (MILP) [50], semi-definite relaxations [40], Lipschitz optimization [6,54], and combining optimization with abstraction [1]. The properties studies include robustness [22,54], reachability (i.e., whether a given output is possible from a given subspace of inputs) [11], and properties expressible with SMT constraints [25,39]. However, these methods cannot in general provide a tight safety bound efficiently or limit in various activation functions. For example, constraint-based approaches such as Reluplex can only work with neural networks with ReLU activation and a few hundreds hidden nodes [25,29,39]. Exhaustive search and global optimisation suffer from the state-space or dimensionality explosion problem [22,41]. Although verification methods used into adversarial training can improve robustness of neural networks, the training process is very inefficient computation. Different from these solutions, the quantification method proposed in this paper can provide a general and efficient framework and even can work efficiently on large-scale neural networks against Lipschitzian properties, as illustrated in Fig. 2. Moreover, we note that while verification is currently working with point-wise robustness, the evidence collected through robustness verification and testing techniques [18,19,47,48] can be utilised to construct safety case for the certification of real-world autonomous systems [65–67].

Safety risks in neural networks

A (feed-forward and deep) neural network can be represented as a function $f : \mathbb{R}^n \to \mathbb{R}^m$, such that given an input $x \in \mathbb{R}^n$, it outputs a probabilistic distribution over a set of m labels $\{1...m\}$, representing the probabilities of assigning labels to the input. We use $f_j(x)$ to denote the probability of labelling an input x with the label j. Based on this, we define the labelling function $l : \mathbb{R}^n \to \{0...m\}$ as

$$l(x) = \begin{cases} k & |f_k(x) - \max_{j \neq k} f_j(x)| > \epsilon, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

where $k = \max_j f_j(x)$ is the label with the greatest confidence and ϵ is a threshold value. Intuitively, if there is a label $k \in \{1...m\}$ with *significant confidence* comparing to other labels $j \neq k$, we assign x with the *label* k. On the other hand, if there is no label with significant confidence comparing to other labels, we assign x with the label 0, denoting that the network is not confident about its own decision.

In practice, a neural network is a complex, highly nonlinear function composed of a sequence of simple, linear, or nonlinear functional mappings [7,15]. Typical functional mappings include fully connected, convolutional, pooling, Softmax, and Sigmoid. In this paper, we treat the network as a black-box, and therefore, our methods can work with any internal layer and any architectures as long as the network is feed-forward.

Safety risk: By training over a labelled dataset, a neural network f is to simulate the decisions of a human $O : \mathbb{R}^n \to \{0...m\}$ on unseen inputs, where $O(x) = 0$ represents that the human cannot decide on its labelling. Therefore, the safety risk of f lies on the *inconsistency* of decisions between $l(x)$ and $O(x)$, as defined in Definitions 1 and 2.

Definition 1 (*Misalignment on Decision*) Given a network $f : \mathbb{R}^n \to \mathbb{R}^m$, a human decision oracle $O : \mathbb{R}^n \to \{0...m\}$, and a legitimate input $x \in \mathbb{R}^n$ such that $l(x) = O(x) \neq 0$, we have Table 1 for $\hat{x}$ being another input that is perturbed from x and the perturbations are small.

Intuitively, each entry in Table 1 represents a possible scenario for input x and $\hat{x}$. For example, those entries on the diagonal represent that no obvious error can be inferred. For the case where $0 \neq l(\hat{x}) \neq l(x)$ and $O(\hat{x}) = O(x)$, human believes that the two inputs are in the same class, but the network believes not, representing a typical case of adversarial example [49]. The two entries with $O(\hat{x}) = 0$ represent the scenarios where human is uncertain about $\hat{x}$, while the network has high confidence about it. They are also

Table 1 Categories of safety risks by the alignment of neural network decisions with human perception uncertainty examples are for the first time studied in this paper

Decisions	$O(\hat{x}) = 0$	$O(\hat{x}) = O(x)$	$0 \neq O(\hat{x}) \neq O(x)$
$l(\hat{x}) = 0$	No error	Uncertainty example	Uncertainty example
$0 \neq l(\hat{x}) = l(x)$	Adversarial example [49]	No error	Invariant example [24]
$0 \neq l(\hat{x}) \neq l(x)$	Adversarial example [49]	Adversarial example [49]	No error

seen as adversarial examples. Moreover, invariant example [24] occurs when x and $\hat{x}$ are labelled as the same, while human believes that they should belong to different classes. Finally, uncertainty example, to be discussed for the first time in this paper, covers two entries where the network is uncertain when human can clearly differentiate.

Uncertainty may lead to safety concern in practice. For example, it has been well discussed that adversarial examples [49] may lead to disastrous consequences. For instance, in a shared autonomy scenario where a human driver relies on a deep learning system to make most of the decisions and expects its handing over of the control only when necessary, the deep learning system may act confidently (i.e., $l(\hat{x}) \neq 0$) when human believes that it should perform the other action (i.e., $O(\hat{x}) = O(x) = l(x) \neq l(\hat{x})$) or ask for the transfer of control back to human (i.e., $O(\hat{x}) = 0$). These are adversarial examples. On the other hand, the uncertainty example suggests the other serious consequence: it is possible that the deep learning system intends to hand back the control (**since** $l(\hat{x}) = 0$), while the human driver believes that the deep learning is able to handle it very well and loses her concentration (cf. Tesla incident and Uber incident[3]).

Besides the risks from the mis-alignment of prediction decisions (i.e., adversarial example, invariant example, and uncertainty example), we have the following:

Definition 2 (*Misalignment on rigidity of classification probability*) Given a probability $f_j(x)$ and a pre-specified constant ϵ, it is possible that human may expect the unreachability of $f_j(x) + \epsilon$ under certain perturbation on x, while neural network can. We call those perturbed inputs $\hat{x}$ that satisfy $f_j(\hat{x}) \geq f_j(x) + \epsilon$ reachability examples.

Norm ball: In Definition 1, we use "$\hat{x}$ being another input that is perturbed from x" to state that $\hat{x}$ is close to x. This is usually formalised with norm ball as follows:

$$\mathcal{B}(x, d, p) = \{\hat{x} \mid ||\hat{x} - x||_p \leq d\}. \tag{2}$$

Intuitively, $\mathcal{B}(x, d, p)$ includes all inputs that are within a certain distance to x. The distance is measured with L_p-norm, such that $||x||_p = (\sum_{i=1}^{n} x_i^p)^{1/p}$. The "certain perturbation on x" in Definition 2 is also formalised in this way.

[3] Refer https://www.bbc.co.uk/news/technology-56799749 and https://www.reuters.com/article/uber-selfdriving-idUSKBN26708P.

Quantification of safety risks

In this paper, we consider three safety risks: adversarial example, uncertainty example, and reachability example. First of all, we take a generic definition of safety property.

Definition 3 A safety property $s(x)$ is an expression over the outputs $\{f_i(x) \mid i \in \{1...m\}\}$ of the neural network, and we expect that whenever $s(x) < 0$, the neural network has safety risk.

In the following, we show how to instantiate $s(x)$ with *specific expressions* to quantify the robustness, the reachability, and the uncertainty.

Robustness quantification of safety risks

First, a norm ball $\mathcal{B}(x, d, p)$ is a safe norm ball if $l(\hat{x}) = l(x)$ for all $\hat{x} \in \mathcal{B}(x, d, p)$. Moreover, a norm ball $\mathcal{B}(x, d, p)$ is a targeted safe norm ball w.r.t. a pre-specified label l if $l(\hat{x}) \neq l$ for all $\hat{x} \in \mathcal{B}(x, d, p)$. Intuitively, a safe norm ball requires all the inputs within it to have the same label as the center point x, while a targeted safe norm ball is to avoid having any input to have a specific label l.

Based on safe norm balls, we define the robustness as below.

Definition 4 (*Robustness*) Given a network f, an input x, and a norm ball $\mathcal{B}(x, d, p)$, the robustness of f on x and $\mathcal{B}(x, d, p)$ is to find the maximum radius d' that can make $\mathcal{B}(x, d', p)$ safe. More specifically, $\mathcal{B}(x, d', p)$ is a safe norm ball, and for all $d'' > d'$, $\mathcal{B}(x, d'', p)$ is not a safe norm ball. We use $R(x, d, p)$ to denote such a maximum safe radius d', and call it robustness radius.

It is noted that $R(x, d, p) \leq d$. Intuitively, the robustness of f on x and $\mathcal{B}(x, d, p)$ is evaluated with the maximum radius of safe norm balls, which are centered at x and within the norm ball $\mathcal{B}(x, d, p)$. We remark that accurately calculating the robustness is extremely difficult in a high-dimensional space; see e.g., [25,49].

Below, we instantiate the safety property $s(x)$ with *Confidence Interval* expression, which can be used to quantify the robustness.

Definition 5 (*Confidence-interval expression*) Let f be a network, x an input, and $l_1, l_2 \in \{1...m\}$ two labels, we define

confidence-interval expression as follows:

$$s_{CI}(x)(l_1, l_2) = f_{l_1}(x) - f_{l_2}(x) - \epsilon, \tag{3}$$

where $\epsilon \in [0, 1]$ specifies the minimum confidence interval required by the user.

According to Definition 3, we use $s(x) < 0$ to express the existence of potential risks. Therefore, intuitively, the expression $s_{CI}(x)(l_1, l_2)$ suggests a safety specification that the confidence gap between labels l_1 and l_2 on input x has to be larger than a pre-specified value ϵ. Depending upon the concrete safety requirements, a user may instantiate l_1, l_2, and ϵ into different values. We can instantiate l_1 and l_2 and obtain the following concrete confidence-interval expressions:

- Case-1: $s_{CI}(x)(j_1, j_2)$, where for some other input $x_0 \neq x$, $j_1 = \arg\max_j f_j(x_0)$ is the label with the greatest confidence value and $j_2 = \arg\max_{j \neq j_1} f_j(x_0)$ is the label with the second greatest confidence value;
- Case-2: $s_{CI}(x)(j_1, l)$ for some given label l;
- Case-3: $s_{CI}(x_0)(j_1, j_m)$, where $j_m = \arg\min_j f_j(x_0)$ is the label with the smallest confidence value.

Intuitively, the above expression maintain different types of discrepancies between two confidence values of an input x. In particular, the expression $s_{CI}(x)(j_1, j_2)$ in Case-1 is closely related to the resistance of DNNs to untarget adversarial attacks. Expression in Case-2 is reflecting the robustness to target adversarial attacks. In both cases, we may use $\epsilon = 0$, to denote a mis-classification, or assign ϵ with some value to make sure that the network mis-classifies with high confidence (a more serious scenario). And expression in Case-3 instead captures the largest variation between confidence values.

While $s_{CI}(x)(j_1, j_2)$ provides an expressible way to specify whether an input *directly* leads to the safety risk, we need to show how to use this expression for the purpose of evaluating robustness. Below, we define a Lipschitzian metric.

Definition 6 (*Lipschitzian Metric*) Given an expression $s(x)$, a norm ball $\mathcal{B}(x, d, p)$ centered at an input x, we let $Q(s, x, d, p)$ be a Lipschitzian metric, defined as follows:

$$Q(s, x, d, p) = \sup_{\hat{x} \in \mathcal{B}(x,d,p)} \frac{|s(x) - s(\hat{x})|}{||x - \hat{x}||_p}. \tag{4}$$

Based on a given point x, the metric is intuitive to find the greatest changing rate within the norm ball $\mathcal{B}(x, d, p)$. The following theorem shows that the robustness radius $R(x, d, p)$ can be estimated *conservatively* if the Lipschitzian metric can be computed.

Theorem 1 *Given a neural network f, an input x, and a norm ball $B(x, d, p)$, we have that $\mathcal{B}(x, d', p)$ is a safe norm ball when* $d' = \dfrac{s(x)}{Q(s, x, d, p)} \leq d$.

Proof By the robustness definition in Definition 4, we need to have

$$\forall \theta : ||\theta||_p \leq d' \Rightarrow s(x + \theta) \geq 0. \tag{5}$$

Since neural networks are Lipschitz [41], we have that for all $x + \theta \in \mathcal{B}(x, d, p)$

$$|s(x) - s(x + \theta)| \leq Q(s, x, d, p)\ ||\theta||_p. \tag{6}$$

We consider two possible cases: $s(x + \theta) \geq s(x)$ or $s(x + \theta) < s(x)$. For the case of $s(x + \theta) \geq s(x)$, it is straightforward that $s(x + \theta) \geq 0$, since $s(x) \geq 0$ by the safety requirement. For the case of $s(x + \theta) < s(x)$, we have that

$$s(x) - Q(s, x, d, p)\ ||\theta||_p \leq s(x + \theta). \tag{7}$$

To ensure $s(x + \theta) \geq 0$, it is sufficient to have $s(x) - Q(s, x, d, p)\ ||\theta||_p \geq 0$. By $||\theta||_p \leq d'$, it is sufficient to have $s(x) - Q(s, x, d, p)\ d' = 0$. Therefore, if we have $d' = s(x)/Q(s, x, d, p)$, then Eq. (5) holds, i.e., $\mathcal{B}(x, d', p)$ is a safe norm ball.

Moreover, we require that $d' \leq d$, since otherwise Eq. (6) may not hold. Intuitively, this is because the computation of $Q(s, x, d, p)$ is conducted within $\mathcal{B}(x, d, p)$, and hence, any result based on it may not work over a greater norm ball. □

The above theorem suggests that we can use $s(x)/Q(s, x, d, p)$ to conservatively estimate the robustness radius $R(x, d, p)$. It is known that $s(x)$ is trivial, so the estimation of robustness radius $R(x, d, p)$ is reduced to the estimation of Lipschitz metric $Q(s, x, d, p)$.

Uncertainty quantification of safety risks

As explained in Definition 1, adversarial examples—the risk for robustness—are not the only class of safety risks. In this section, we study another type of safety risk, i.e., uncertainty examples. To the best of our knowledge, this is the first time this safety risk is studied. We remark that the study of this risk becomes easy, owing to our approach of taking a generic expression $s(x)$. Also, its estimation and detection can take the same algorithm as the robustness quantification. That is, it comes for free.

Since uncertainty examples represent those inputs on which the network f cannot have a clear decision, we need to express the *uncertainty* of the distribution $f(x)$. This can be done by considering the Kullback–Leibler divergence (or KL

divergence) [38] from $f(x)$ to e.g., the uniform distribution or another distribution $f(\hat{x})$.

Definition 7 (Uncertainty expression) Let f be a network and x an input, and we write

$$s_U(x) = -\epsilon - \sum_{l=1}^{m} \frac{1}{m} \log m f_l(x), \tag{8}$$

where $\epsilon > 0$ is a bound representing, from the DNN developer's view, what is the smallest KL divergence from the uniform distribution for input x to be classified as a good behaviour. $f_l(x)$ denotes the probabilistic confidence of label l where $l \in \{1, 2, \ldots, m\}$ given an input x. For example, the MNIST dataset has ten distinguishing labels, so $m = 10$. Moreover, if consider the other distribution $f(\hat{x})$ as the basis, we can have a generalized uncertainty expression

$$s_U(x, \hat{x}) = -\epsilon - \sum_{l=1}^{m} f_l(\hat{x}) \log \frac{f_l(x)}{f_l(\hat{x})}. \tag{9}$$

Intuitively, the uniform distribution indicates that the network is unsure about the input. Therefore, in Eq. (8), we require as a necessary condition, for the decision on x to be safe, that the KL divergence from $f(x)$ to the uniform distribution, the express $(-\sum_{l=1}^{m} \frac{1}{m} \log m f_l(x))$, is greater than ϵ. If so, it is believed that the network behaves well on the input x. We remark that the computation of uncertainty example of this kind can be difficult, because it lies on the vicinity of the intersection point of all decision boundaries (as illustrated in Fig. 1) and such areas are sparse in the input space.

Moreover, $s_U(x, \hat{x})$ requires that the decision of x is significantly far away from $\hat{x}$. That is, it allows a *user-defined safety risk* $f(\hat{x})$ and asks for the network decision to stay away from the risks.

Based on the expressions, we can also define safe norm balls by requiring that no input in a norm ball satisfies $s(x) < 0$. The definition of maximal safe norm ball can also be extended to this context, and we can define the uncertainty metric the same as that of Definition 6. Without loss of generality, we will continue use $\mathcal{B}(x, d, p)$ and $Q(s, x, d, p)$ to denote them, respectively. In fact, $Q(s, x, d, p)$ for uncertainty quantification is based on $s_U(x)$, which exactly is equal to $Q_U(s, x, d, p)$, as showed in Fig. 2. As before, a conservative estimation of the maximum radius $\mathcal{B}(x, d, p)$ of safe norm balls can be reduced to the computation of $Q(s, x, d, p)$. Therefore, *the study of uncertainty quantification comes for free if we are able to work with the robustness quantification*.

Reachability quantification of safety risks

For reachability, we can define the following expression: $s_R(x)(l) = f_l(x) - \epsilon$, where $\epsilon \in (0, 1)$ is a pre-specified threshold for the rigidity of classification probability. Other notions, such as $\mathcal{B}(x, d, p)$ and $Q(s, x, d, p)$, follow the discussion in "Robustness quantification of safety risks". Based on reachability expression $s_R(x)$, we can evaluate reachability $Q_R(s, x, d, p)$ by Lipschitzian metric.

Safety risk quantification algorithms

Based on the various safety expression $s(x)$, we present a general framework DeepQuant for quantifying three safety risks. To calculate the Lipschitzian metric $Q(s, x, d, p)$ as in Definition 6, we consider practical method rather than using gradient-based adversarial attack or the formal analysis via encoding of neural networks—as we discussed in the related work ("Related work"). A derivative-free optimisation method is proposed. This method can efficiently search over samples in norm ball $\mathcal{B}(x, d, p)$. We remark that we use robustness—$Q(s, x, d, p)$ and $\mathcal{B}(x, d, p)$—as example, and the algorithms can work with both uncertainty and reachability.

Given a trained DNN f, a property expression $s : \mathbb{R}^m \to \mathbb{R}$, and a genuine $x \in \mathbb{R}^n$, the Lipschitzian metric can be calculated by solving the following optimization problem:

$$\min_{\hat{x}} \quad w(\hat{x}) \quad s.t. \quad ||\hat{x} - x||_p \leq d \quad \text{and} \quad \hat{x} \in [0, 1]^n, \tag{10}$$

where $w(\hat{x}) = ||\hat{x} - x||_p / |s(\hat{x}) - s(x)|$. The optimization problem contains a non-convex objective (due to the non-convexity of DNNs), together with a set of constraints. Note that, for $p \in \{1, 2\}$, the constraints include both nonlinear inequality constraints and box-constraints, and for $p = \infty$, the constraints include only with box-constraints.

The optimization is based on a composition of the DNN f and the property expression s, both of which may be non-differential or not smooth. The analytic form of its first-order derivative is also difficult to get. Methodologically, to achieve the broadest applications, we need a single optimization method that can efficiently estimate different DNN properties for various property expressions regardless its differentiability, smoothness, or whether an analytic form of derivative exits. In this regard, instead of using gradient-based method, we take a derivative-free optimization framework. Our optimization solutions are centered around the mesh adaptive direct search (MADS) [4], which is designed for black-box optimization problems for which the functions defining the objective and the constraints are typically seen as black-

boxes [5]. It requires no gradient or derivative information but still provides a convergence guarantee to the first-order stationary points based on the Clarke calculus [3–5].

In the following, we will present an algorithm for L_∞ norm ("L_∞-norm risk quantification"), enhance the algorithm with tensor-based parallelisation for GPU implementation ("Tensor-based parallelisation for L_∞-norm risk quantification"), and present an algorithm for L_1 and L_2 norm ("L_1 and L_2-norm risk quantification").

L_∞-norm risk quantification

First, we introduce MADS in the context of risk quantification based on L_∞-norm. When $p = \infty$, we can transform Eq. (10) into the following problem:

$$\min_{\hat{x}} \quad w(\hat{x}) \quad s.t.\ l_d \leq \hat{x} \leq u_d, \tag{11}$$

where lower bound is $l_d = \max\{x - d, 0\}$ and upper bound is $u_d = \min\{x + d, 1\}$. Instead of presenting the details of MADS [4], we give its idea. Briefly, MADS seeks to improve the current solution by testing points in the neighborhood of the current point (the *incumbent*). Each point is one step away in one direction on an iteration-dependent mesh. In addition to these points, MADS can incorporate any search strategy into the optimization to have additional test points. The above process iterates until a stopping condition is satisfied.

Formally, each iteration of MADS comprises of two stages, a SEARCH stage and an optional POLL stage. The SEARCH stage evaluates a number of points proposed by a given search strategy, with the only restriction that the tested points lie on the current mesh. The current mesh at the kth iteration is $M_k = \bigcup_{x \in S_k} \left\{x + \Delta_k^{\text{mesh}} z \mathbf{D}^{(i)} | z \in \mathbb{N}, \mathbf{D}^{(i)} \in \mathbf{D}\right\}$, where $S_k \subset \mathbb{R}^n$ is the set of points evaluated since the start of the iteration, $\Delta_k^{\text{mesh}} \in \mathbb{R}_+$ is the *mesh size*, and $\mathbf{D}$ is a fixed matrix in $\mathbb{R}^{n \times n_\mathbf{D}}$ whose $n_\mathbf{D}$ columns represent viable search directions. We let $\mathbf{D}^{(i)}$ be the ith column of $\mathbf{D}$. In our implementation, we let $\mathbf{D} = [\mathbf{I}_n, -\mathbf{I}_n]$, where $\mathbf{I}_n$ is the n-dimensional identity matrix.

The POLL stage is performed if the SEARCH fails in finding a point with an improved objective value. POLL constructs a *poll set* of candidate points, P_k, defined as $P_k = \left\{x_k + \Delta_k^{\text{poll}} \mathbf{D}^{(i)} | \mathbf{D}^{(i)} \in \mathbf{D}_k\right\}$, where x_k is the incumbent and $\mathbf{D}_k$ is the set of *polling directions* constructed by taking discrete linear combinations of the set of directions $\mathbf{D}$. The *poll size* parameter $\Delta_k^{\text{poll}} \geq \Delta_k^{\text{mesh}}$ defines the maximum length of poll displacement vectors $\Delta_k^{\text{mesh}} \mathbf{D}^{(i)}$, for $\mathbf{D}^{(i)} \in \mathbf{D}_k$ (typically, $\Delta_k^{\text{poll}} \approx \Delta_k^{\text{mesh}} \|v\|$). Points in the poll set can be evaluated in any order, and the POLL is opportunistic in that it can be stopped as soon as a better solution is found. The POLL stage ensures theoretical convergence to a local stationary point according to Clarke calculus for nonsmooth functions [5].

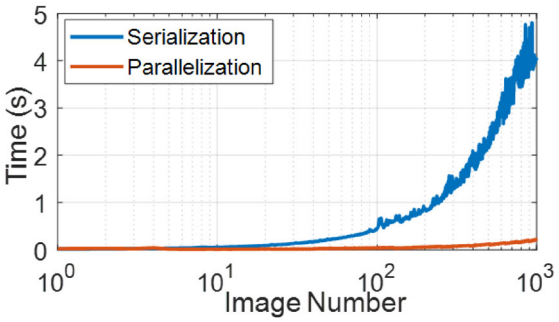


Fig. 3 Number of queries to the DNN w.r.t. the number of images, with and without tensor-based parallelization—a significant motivation for our tensor-based parallelisation algorithm

If either SEARCH or POLL succeeds in finding a mesh point with an improved objective value, the incumbent is updated and the mesh size remains the same or is multiplied by a factor $\tau > 1$. If neither SEARCH nor POLL is successful, the incumbent does not move and the mesh size is divided by τ. The algorithm proceeds until a stopping criterion is met (e.g., maximum budget of function evaluations).

Tensor-based parallelisation for L_∞-norm risk quantification

For the problem as in Eq. (10), objective function $w(\hat{x})$ includes neural network $f(\hat{x})$. Given the availability of tensor-based algorithmic operations in deep learning frameworks such as TesnorFlow, PyTorch, and Caffe, etc., we improve the algorithm described in "L_∞-norm risk quantification" with a tensor-based parallelization, so as to achieve computational efficiency with GPU. As shown in Fig. 3, with a low-end Nvidia GTX1050Ti GPU, to evaluate a 16-layer MNIST DNN on 1000 images, the time using tensor-based parallelization is 25 times faster than without using one. Specifically, our new algorithm— enhancing MADS with parallelization—can improve the speed roughly $(n_k + m_k)/2$ times in terms of DNN inquiry numbers, where n_k and m_k—to be introduced below—are such that n_k is around $\geq 2n$ depends on the search strategy and iterations and m_k is at least $\geq n + 1$.

Comparing to the traditional MADS in [4], we perform the following improvements in terms of parallelization in both SEARCH and POLL stages. Algorithm 1 provides the pseudo-code for the Parallelised algorithm.

– Parallelisation in SEARCH Stage: Assuming at kth iteration, there are n_k hyper-points, i.e., $\{x_1^k, x_2^k, \ldots, x_{n_k}^k\} \in M_k$, We stack all those hyper-points into a 3-D Tensor $\mathcal{M}^k$ such that $\mathcal{M}^k(i, j, k)$ is the ith element in x_j^k. Then,

Algorithm 1: Tensor-based parallelised MADS (TP-MADS)

Input: Objective function $w(x)$, starting point x_0, variable constraint l_d and u_d
Initialization: $\Delta_0^{\text{mesh}} \leftarrow 2^{-10}$, $\Delta_0^{\text{poll}} \leftarrow 1$, $k \leftarrow 0$, evaluate $w(x)$ on initial design
while `fevals` $>$ `MaxFunEvals` ***or*** $\Delta_k^{\text{poll}} < 10^{-6}$ **do**
  Stack $\{x_1^k, x_2^k, ..., x_{n_k}^k\} \in M_k$ into a tensor format $\mathcal{M}^k$;
  Evaluate $w(x)$ on $\mathcal{M}^k$ via parallelization;
  if SEARCH *is* NOT successful **then**
    Stack $P_k = \{x_1^k, x_2^k, ..., x_{m_k}^k\}$ into a tensor format $\mathcal{P}^k$;
    Evaluate function $w(x)$ on $\mathcal{P}^k$ via parallelization;
  end
  if *Iteration k is* successful **then**
    Update incumbent x_{k+1};
    if POLL *was* successful **then**
      $\Delta_k^{\text{mesh}} \leftarrow 2\Delta_k^{\text{mesh}}$, $\Delta_k^{\text{poll}} \leftarrow 2\Delta_k^{\text{poll}}$;
    else
      $\Delta_k^{\text{mesh}} \leftarrow \frac{1}{2}\Delta_k^{\text{mesh}}$, $\Delta_k^{\text{poll}} \leftarrow \frac{1}{2}\Delta_k^{\text{poll}}$
    end
  end
  Update $k \leftarrow k + 1$
end
Output: $x_{\text{end}} = \arg\min_k w(x_k)$ and $w(x_{\text{end}})$

we feed $\mathcal{M}^k$ into the GPU to perform the DNN evaluation.

– Parallelisation in POLL Stage: Assuming at kth iteration, there are m_k points in set $P_k = \{x_1^k, x_2^k, \ldots, x_{m_k}^k\}$. We stack all those hyper-points into a 3-D Tensor $\mathcal{P}^k$, such that $\mathcal{P}^k(i, j, k)$ is the ith element in x_j^k. Then, we feed $\mathcal{P}^k$ into the GPU to perform the DNN evaluation.

L_1 and L_2-norm risk quantification

For L_1 or L_2-norm, we need to solve an optimization problem with box-constraint as well as nonlinear inequality constraints, as shown in Eq. (10). We take an augmented Lagrangian algorithm [27] to solve a nonlinear optimization problem with nonlinear constraints, linear constraints, and bounds. Specifically, bounds and linear constraints are handled separately from nonlinear constraints. We transform the constrained optimization problem into an unconstrained problem by combining the fitness function and nonlinear constraint function using the Lagrangian and the penalty parameters, as below

$$\boldsymbol{\Theta}(x, \lambda, s) = w(x) - \lambda q \log(q + c(x)), \tag{12}$$

where $x \in [0, 1]^n$, $\lambda > 0$ is a Lagrange multiplier, $q > 0$ is a positive shift, and $c(x) = ||x - x_0||_p - d$ where $p \in \{1, 2\}$.

Algorithm 2 provides the pseudo-code to solve the L_1 and L_2-norm risk quantification problem. The idea of the algorithm is as follows. It starts by initialising parameters λ and q. Then, we minimise a sub-problem, which has fixed values for λ and q and is solved by calling Tensor-based Parallelised Mesh Adaptive Direct Search, as shown in Algorithm 1. When the sub-problem is minimised to a required accuracy and satisfies feasibility conditions, the Lagrangian estimate (Eq. (12)) is updated. Otherwise, the penalty parameter λ is increased by a penalty factor, together with an update on q. This results in a new sub-problem formulation and minimization problem. The above steps (other than the initialization) are repeated until a stopping criteria is met.

Algorithm 2: TP-MADS with inequality constraints

Input: Objective function $w(x)$, starting point x_0, inequality constraint function $c(x)$, variable constraint $l_d = 0$ and $u_d = 1$
Initialization: Initialize q and λ
while *Termination criteria not satisfied* **do**
  Call for Algorithm-1 to solve a Sub-problem Eqn. (12);
  Update Lagrange multiplier estimate λ;
  Update positive shift q;
end
Output: $x_{\text{end}} = \arg\min_{x,\lambda,q} \boldsymbol{\Theta}(x, \lambda, q)$ and $w(x_{\text{end}})$

Experimental results

First, in "Experiments on reachability quantification", by comparing with state-of-the-art tools on reachability quantification, we show the efficiency of DEEPQUANT. Then, in "Experiments on robustness quantification", by conducting robustness quantification on networks of different scales, over datasets MNIST, CIFAR-10, and ImageNet, we show the **tightness** of results and the scalability of DEEPQUANT. Finally, in "Experiments on uncertainty quantification", we conduct experiments on uncertainty quantification.[4]

Experiments on reachability quantification

Three state-of-the-art tools are considered. Reluplex [25] is an SMT-based method for DNNs with ReLU activations; we apply a bisection scheme to achieve the reachability quantification. SHERLOCK [11] is an MILP-based method dedicated to reachability quantification on DNNs with ReLU activations. DeepGO [41] is a general reachability quantification tool that can work with a broad range of neural networks including those with non-ReLU activation layers.

We followed the experimental setup in [11] and trained ten networks, including six ReLU networks and four Tanh networks (i.e., networks with tanh activations). Note that neither SHERLOCK nor Reluplex can work with Tanh networks (i.e., tanh-NN-6 to tanh-NN-9). For ReLU networks,

[4] The software will be found at https://github.com/TrustAI/DeepQuant.

Table 2 Comparison with SHERLOCK [11], Reluplex [25], and DeepGO [41]

NN ID	Layer×Neuron	SHERLOCK	Reluplex	DeepGO	DEEPQUANT
ReLU-NN-0	1 × 100	1.9 s	1 m 55 s	0.4 s	1.80 s
ReLU-NN-1	1 × 200	2.4 s	13 m 58 s	1.0 s	1.56 s
ReLU-NN-2	1 × 500	17.8 s	Timeout	6.8 s	**1.21 s**
ReLU-NN-3	1 × 500	7.6 s	Timeout	5.3 s	**1.26 s**
ReLU-NN-4	1 × 1000	7 m 57.8 s	Timeout	1.8 s	**1.21 s**
ReLU-NN-5	6 × 250	9 m 48.4 s	Timeout	15.1 s	**2.81 s**
tanh-NN-6 (2-input)	6 × 250	N/A	N/A	14.8 s	**2.93 s**
tanh-NN-7 (3-input)	6 × 250	N/A	N/A	58.7 s	**8.92 s**
tanh-NN-8 (4-input)	6 × 250	N/A	N/A	394.1 s	**20.94 s**
tanh-NN-9 (5-input)	6 × 250	N/A	N/A	2680.4 s	**129.81 s**

To highlight our method (DeepQuant) is more efficient than baseline methods

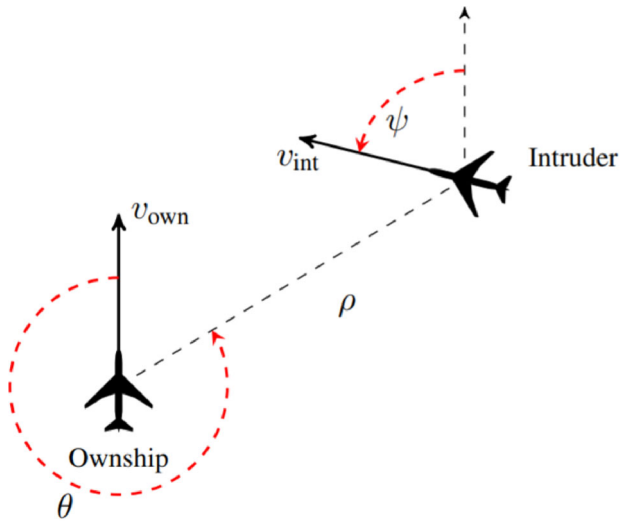


Fig. 4 Geometry for ACAS-Xu horizontal logic table (from [25])

i.e., ReLU-NN-0 to ReLU-NN-5, the input has two dimensions, i.e., $x \in [0, 10]^2$. The input dimensions for tanh-NN-6 to tanh-NN-9 are gradually increased, from $x \in [0, 10]^2$ to $x \in [0, 10]^5$. For fairness of comparison, we implement DEEPQUANT in Matlab2018a, running on a Laptop with i7-7700HQ CPU and 16GB RAM. The software and hardware setup are made exactly the same as DeepGO [41]. Both Reluplex[5] and SHERLOCK[6] are configured to run on a different software platform and a more powerful hardware platform—a Linux workstation with 63GB RAM and a 23-Core CPU. We record the running time of each tool when its reachability error is within 10^{-2}. The comparison results are given in Table 2.

From Table 2, DEEPQUANT is consistently better than SHERLOCK and Reluplex. For the six ReLU-based networks, DEEPQUANT has an averaged computation time of around 1.6s, which has *108-fold* and *300-fold* improvement over SHERLOCK and Reluplex (excluding timeouts), respectively. Furthermore, *the performances of both Reluplex and SHERLOCK are considerably affected by the increase of neuron numbers and layers, while* DEEPQUANT *does not*. Although both DeepGO and DEEPQUANT can work on Tanh networks, DeepGO is significantly more sensitive to the dimension of the input space, with the computation time is nearly exponential w.r.t. the input dimension. Thus, *for a neural network with high-dimensional inputs,* DEEPQUANT *demonstrates significant superiority over DeepGO*. For example, for the neural network tanh-NN-9 (with five input dimensions), DEEPQUANT is nearly *20* times faster.

In summary, DEEPQUANT exhibits better efficiency over other algorithms such as Reluplex and SHERLOCK by its computational complexity measurement. Namely, the computational complexity of DEEPQUANT is NP over the number of input dimensions, while Reluplex and SHERLOCK are NP over the number of neuron numbers and layers. This leads to clear benefit that DEEPQUANT does not rely on the size of the network and therefore scales much better. Comparing to DeepGO, DEEPQUANT is tensor-parallelized and takes a more sophisticated optimization algorithm.

Experiments on robustness quantification

ACSC-Xu networks

The first experiment is performed on a 5-input and 5-output ACSC-Xu neural networks [25]. We aim to validate the accuracy—or tightness—of DEEPQUANT on robustness quantification. From this section, all experiments are conducted on a PC with i7-7700HQ CPU, 16GB RAM, and GPU GTX1050Ti. DNNs are trained with the Neural Network Toolbox in MATLAB2018a. The ACSC-Xu neural network is trained on a simulated dataset and includes 5 fully connected layers, ReLU activation functions, and overall, it contains 300 hidden neurons [25]. The five input variables

[5] https://github.com/guykatzz/ReluplexCav2017.

[6] https://github.com/souradeep-111/sherlock.

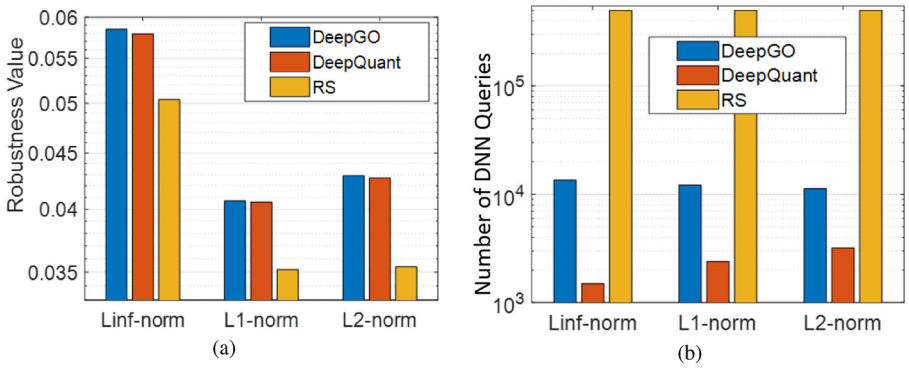


Fig. 5 **a** Accuracy comparison of robustness quantification for L_∞, L_1 and L_2-norm on the ACSC-Xu network. **b** Comparing DNN inquiry numbers when using different methods for robustness quantification on the ACSC-Xu network

Table 3 Structure of MNIST DNN

Layer type	Number of channels	Filter size	Stride value	Activation	Output size
Conv1	1	$3 \times 3 \times 16$	1	ReLU	$28 \times 28 \times 16$
Conv2 + batch	16	$3 \times 3 \times 32$	1	ReLU	$28 \times 28 \times 32$
Conv3 + batch	32	$3 \times 3 \times 64$	1	ReLU	$28 \times 28 \times 64$
Conv4 + batch	64	$3 \times 3 \times 128$	1	ReLU	$28 \times 28 \times 128$
Dropout	–	–	–	–	$28 \times 28 \times 128$
FC	–	–	–	ReLU	256
Dropout	–	–	–	–	256
FC	–	–	–	Softmax	10

The pipeline consists of convolution layer (Conv), batch-normalization layer (Batch), and fully connected layer (FC)

Table 4 Structure of CIFAR-10 DNN

Layer type	Number of channels	Filter size	Stride value	Activation	Output size
Conv1	3	$3 \times 3 \times 32$	1	ReLU	$32 \times 32 \times 32$
Conv2	32	$3 \times 3 \times 32$	1	ReLU	$30 \times 30 \times 32$
MaxPool	32	$2 \times 2 \times 32$	2	–	$29 \times 29 \times 32$
Conv3	32	$3 \times 3 \times 64$	1	ReLU	$29 \times 29 \times 64$
Conv4	64	$3 \times 3 \times 64$	1	ReLU	$27 \times 27 \times 64$
MaxPool	64	$2 \times 2 \times 64$	2	–	$26 \times 26 \times 64$
Dropout	–	–	–	–	$26 \times 26 \times 64$
FC	–	–	–	ReLU	512
FC	–	–	–	Softmax	10

The pipeline consists of convolution layer (Conv), max pooling (MaxPool), and fully connected layer (FC)

of ACAS-Xu neural network are shown in Fig. 4 (which are obtained from various kinds of sensors [9]), where ρ (m) presents distance from ownship to intruder, θ (rad) is angle to intruder relative to ownship heading direction, ψ (rad) shows Heading angle of intruder relative to ownship, and v_{own} (m/s) and v_{int} (m/s) display Speed of ownship and intruder, respectively.

We adapt the safety verification tool DEEPGO [41] for the computation of ground-truth robustness quantification values. Moreover, we implement the other baseline method—a random sampling (RS) method, which uniformly samples 5×10^5 images in a given norm ball.

Figure 5 presents the comparison on the accuracy and the query number, respectively, over different norm distance

Table 5 Detailed information about MNIST and CIFAR-10 dataset

Dataset	Training set size	Testing set size	Testing accuracy	Parameter optimization setup
MNIST	60,000	10,000	99.41%	Max Epochs = 35, Batch = 128, Optimizer = SGDM
CIFAR-10	50,000	10,000	78.30%	Epochs = 50, Batch = 128, Optimizer = SGD

(L_∞, L_1 and L_2). We see that DEEPQUANT can almost reach the ground-truth accuracy value computed by DeepGO (as in Fig. 5a), but with much less number of queries (as in Fig. 5b). Precisely, DEEPQUANT takes around 2×10^3 DNN queries, while DEEPGO requires around 1.3×10^4 DNN queries—6 times difference. Moreover, DEEPQUANT performs much better than RS, on both the tightness and the efficiency. In other word, this experiment exhibits **both the tightness of the result and the efficiency of the computation**.

MNIST and CIFAR-10 networks

We train a 9-layer DNN on MNIST dataset and a 10-layer DNN on CIFAR-10 dataset. Tables 3 and 4 present the model structures of MNIST DNN and CIFAR-10 DNN respectively. Table 5 displays the detail information about training dataset and training parameter setups on MNIST and CIFAR-10.

Figure 6 shows the robustness quantification results for L_1, L_∞-norm and L_2-norm, respectively, on 10 input images (selected from testing dataset) for the MNIST network. The norm balls for these three different robustness quantification are set as $d = 250$, $d = 0.3$, and $d = 8$, respectively. For random sampling, we sampled 1,000,000 images in the norm ball to evaluate $Q(s, x, d, p)$ based on Definition 6. We can see that DEEPQUANT performs consistently better while using tens of times less DNN queries. Please note that, in this experiment, DeepGO is not included due to its limitation on scalability. From Fig. 6, we can see that the proposed robustness quantification method is consistently better than random sampling. Moreover, in our experiment, even through random sampling approach samples 10^6 images, it still cannot achieves an accurate robustness evaluation.

We might also be interested in targeted robustness quantification, which essentially measures the hardness of fooling input images into a given target label. For the CIFAR-10 network, Fig. 7a gives the evaluation results for label-1 as the target label. We can see that label-3 is the most robust, while label-7 is the least robust.

Moreover, Fig. 8 gives some images returned by DEEPQUANT (i.e., $\hat{x}$ in Eq. (4) while evaluating the robustness of MNIST and CIFAR-10 networks. The MNIST images in Fig. 8a are generated when performing L_∞, L_1 and L_2-norm robustness evaluation. The CIFAR-10 images in Fig. 8b are images found by DEEPQUANT when gradually increasing the norm-ball radius (i.e., d in $Q(s, x, d, p)$) from 0.1 to 0.4. It shows that the visual difference w.r.t. input image becomes more obvious for a larger d due to the monotonicity of local robustness value w.r.t. the norm-ball radius. Those images essentially exhibit where the confidence interval decreases the fastest in their corresponding norm balls. We remark that **they are different from adversarial examples, and showcase potentially important robustness risks of a network**.

That is, DEEPQUANT can be used to study variants of safety properties.

ImageNet networks

In Fig. 7b, we measure the robustness of five ImageNet models, including AlexNet (8 layers), VGG-16 (16 layers), VGG-19 (19 layers), ResNet50 (50 layers), and ResNet101 (101 layers), on a L_∞-norm ball for a chosen feature (i.e., a 50×50 square). We can see that, for this local norm space and the chosen feature, ResNet-50 achieves best robustness and AlexNet is the least robust one. This experiment shows the scalability of DEEPQUANT in working with large-scale networks.

In addition, we also present a case study showing how to use DEEPQUANT to guide the Design of Robust DNN Models by using robustness quantification. Figure 9 mainly includes convolution layer (conv), batch normalization layer (batchnorm), and fully connected layer (fc). The DNNs range from with shallow layers (e.g., DNN-1) to deep layers (e.g., DNN-6). We randomly choose 100 images and use DEEPQUANT to evaluate their L_∞-norm robustness. Table 6 presents the result of five input images and the mean robustness values. Based on the robustness statistics, a DNN builder can choose suitable DNNs for different tasks with balance of accuracy and robustness. For example, for a non-critical application that requires high accuracy, DNN-6 is the most suitable one; for a safety-critical application, DNN-2 is a good choice; DNN-4 and DNN-5, however, have good balance on accuracy and robustness.

Experiments on uncertainty quantification

We adopt the same MNIST and CIFAR-10 networks as those in "Experiments on robustness quantification". The detailed experimental setup can be found in Table 5.

In Fig. 10a, we first showcase what is an uncertainty example. The top row is for a true image which has a high KL divergence to uniform distribution, and the bottom is for the uncertainty image found by DEEPQUANT in a L_∞-norm ball ($d = 0.4$). From human perception, the uncertainty image should have the same label as the original one.

In Fig. 10b, we use DEEPQUANT to quantify uncertainty for the six MNIST networks (see Fig. 9 for the details of their structures), while gradually increasing norm-ball radius from 0 to 0.5. We see that the uncertainty of networks vastly worsens with the increase of norm-ball radius. At $d = 0.15$, DNN-1 and DNN-2 show worse uncertainty than other networks. Figure 10c gives some uncertainty examples captured by DEEPQUANT on MNIST and CIFAR-10 neural networks.

In Fig. 11, we visualise several intermediate images obtained during a search for an uncertainty image in a L_∞-norm ball with $d = 0.1$. From left to right, the true image

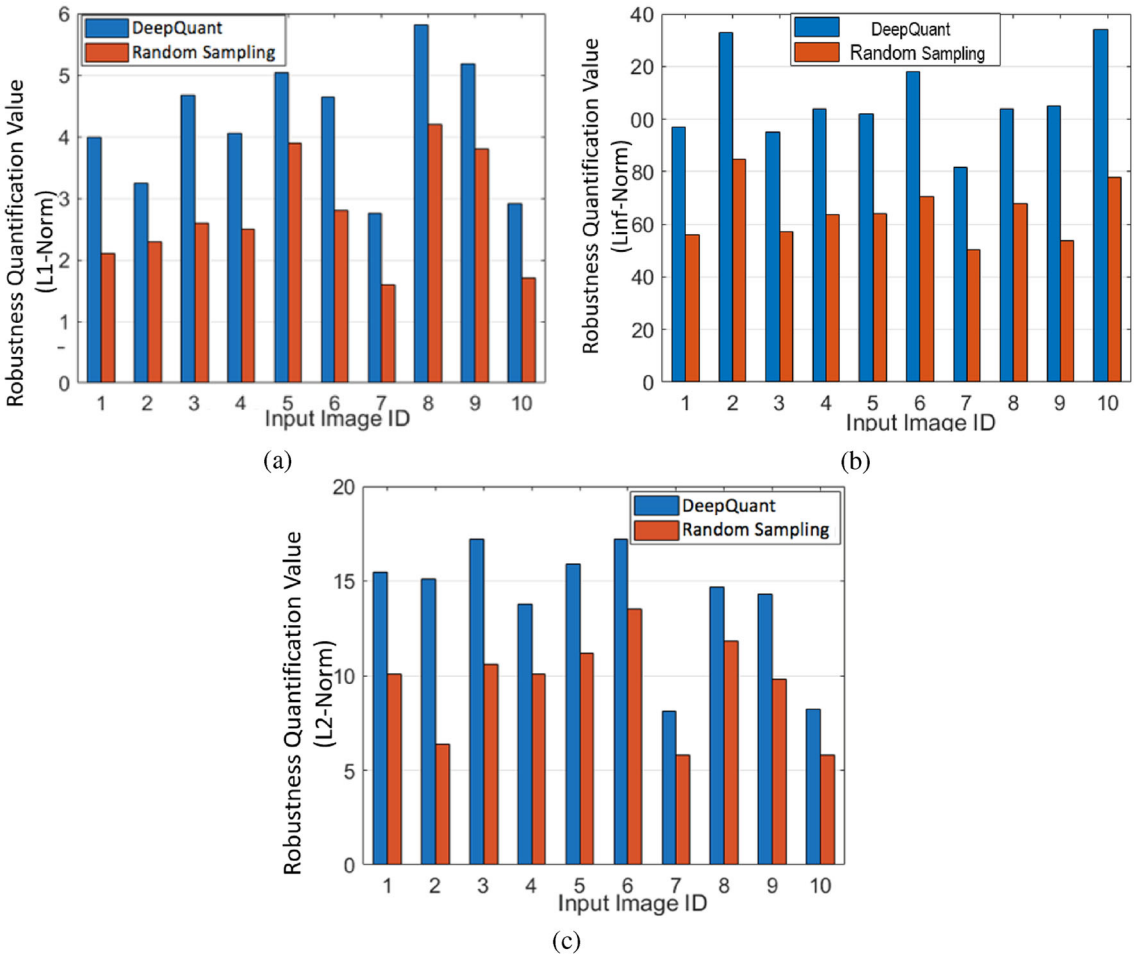


Fig. 6 **a** Comparison of L_1-norm robustness quantification on an MNIST deep neural network, $d = 250$. **b** Comparison of L_∞-norm robustness quantification on an MNIST deep neural network, $d = 0.3$. **c** Comparison of L_2-norm robustness quantification on an MNIST deep neural network, $d = 8$

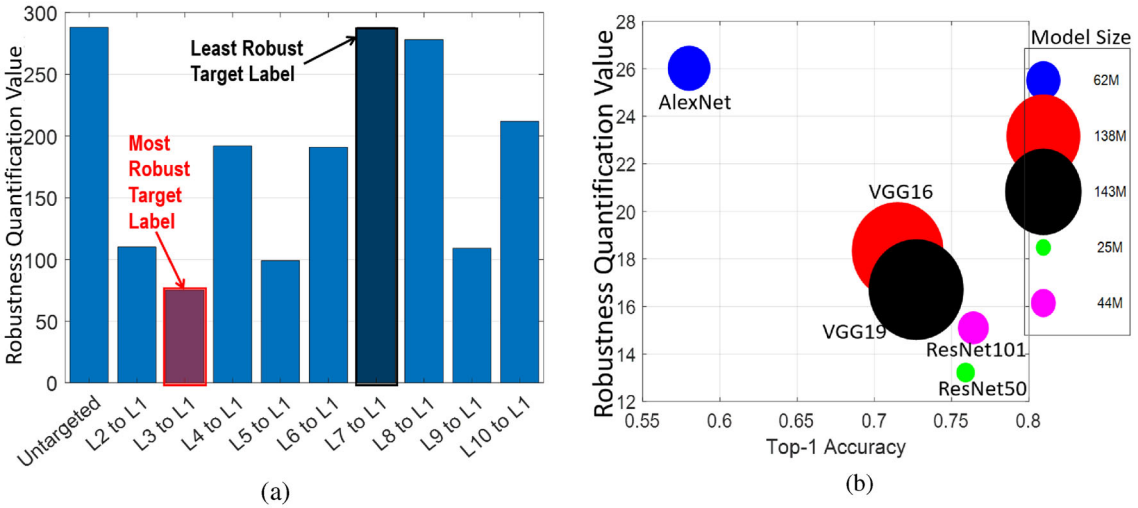


Fig. 7 **a** Targeted robustness evaluation using DEEPQUANT on a CIFAR-10 DNN, with label-1 as the target label. We use LX-L1 on the X-axis to indicate the targeted robustness value from label-X to label-1. **b** Comparison of robustness values of five ImageNet DNNs on an input image using DEEPQUANT

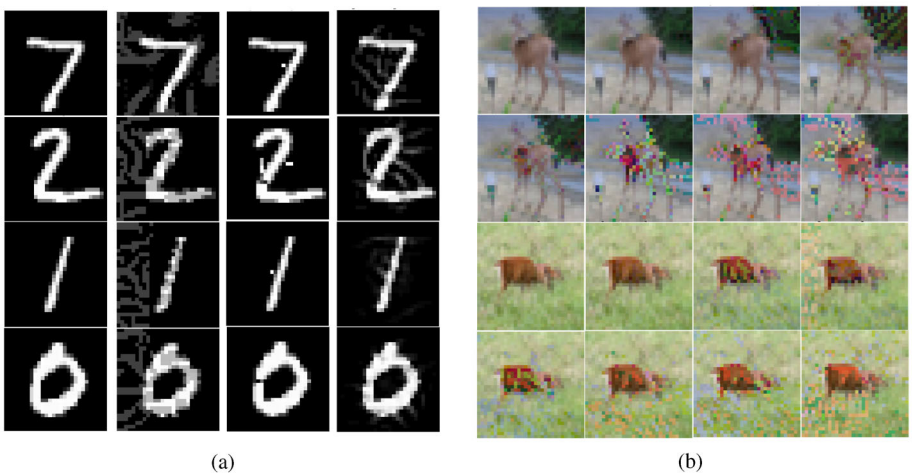


Fig. 8 **a** MNIST images in the first two rows are examples returned by DEEPQUANT, i.e., $\hat{x}$ in Eq. (4). *From Left to Right: it indicates the original image, images using L_∞-norm, L_1-norm, and L_2-norm robustness metric.* **b** CIFAR-10 images in the last two rows are examples returned by DEEPQUANT for an input image by increasing the L_∞-norm-ball radius $d = 0.1 : 0.05 : 0.4$

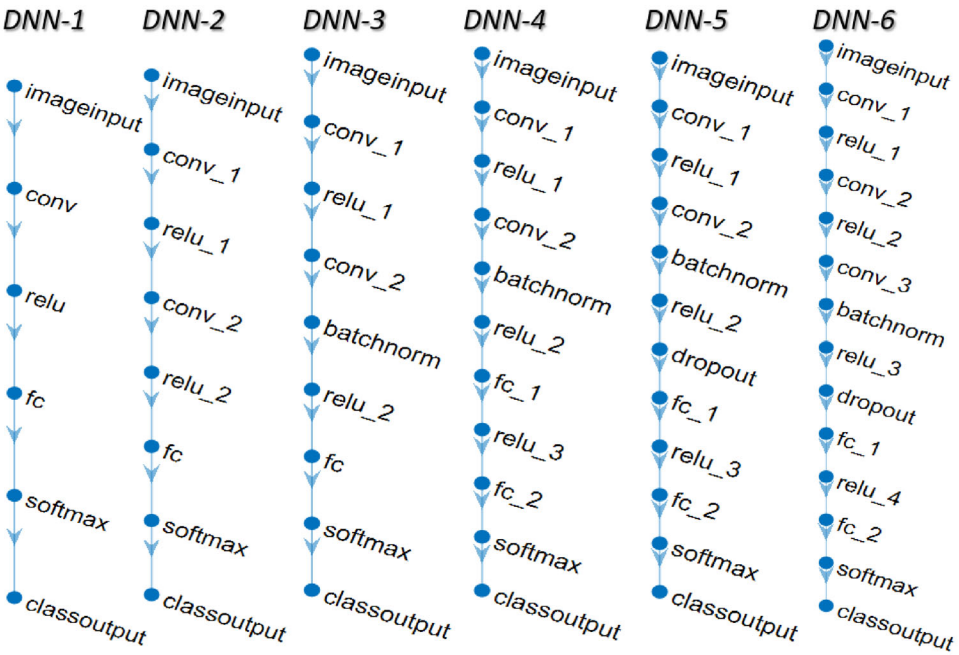


Fig. 9 Model structures of MNIST DNNs from DNN-1 to DNN-6. The filter size of conv_1, conv_2 and conv_3 are $3 \times 3 \times 32$, $3 \times 3 \times 64$, and $3 \times 3 \times 128$, respectively. The probability of dropout is 0.5

is perturbed by DEEPQUANT with an optimization objective of minimising the KL divergence. With the perturbations, the generated images have gradually increased uncertainty related to this specific input. When the KL divergence is reduced to 0, the network is completely confused and does not know how to classify the uncertainty example. Thus, DEEPQUANT is the very first tool that can insightfully and automatically reveal this new, yet very important, safety property in the decision process of a network.

Discussion

To evaluate the safety risks of neural networks, we proposed a new method—DEEPQUANT, which can compute the maximum safe norm ball based on L_p norm and provide theoretical guarantee for DNN models. Leveraging Lipschitz metric and black-box optimization technique, we design a series of optimization algorithms to quantify various safe properties. For more details, you can refer our framework in Fig. 2.

When quantifying reachability, our proposed method DEEPQUANT has significant improvement up to 20 times speed compared with DeepGO, which is very sensitive to the dimension of input space. With the increasing number of neurons and layers, Releplex and SHERLOCK become very inefficient and even timeout. Moreover, using tensor parallization in SEARCH Stage and POLL Stage can improve the efficiency significantly compared with serialization method in Fig. 3. For robustness quantification, DEEPQUANT can achieve comparable accuracy with DeepGo, but only require 1/6 times queries on ACSC-Xu network based on L_∞, L_1 and L_2 norm ball. Due to the computational complexity, DeepGO

Table 6 L_∞-norm robustness quantification results on six MNIST DNNs given five input images

	Img-1	Img-2	Img-3	Img-4	Img-5	Mean	Test Acc.
DNN-1	45.90	93.73	30.44	39.76	93.33	60.63	97.75%
DNN-2	**35.24**	**21.66**	**19.79**	**26.82**	**57.30**	**32.16**	97.95%
DNN-3	87.13	69.40	78.31	84.71	100.30	83.97	98.38%
DNN-4	42.07	46.42	69.90	46.86	63.42	53.73	99.06%
DNN-5	53.68	54.17	82.78	41.21	65.75	59.52	99.16%
DNN-6	96.91	70.13	75.53	64.63	80.19	77.48	**99.41**%

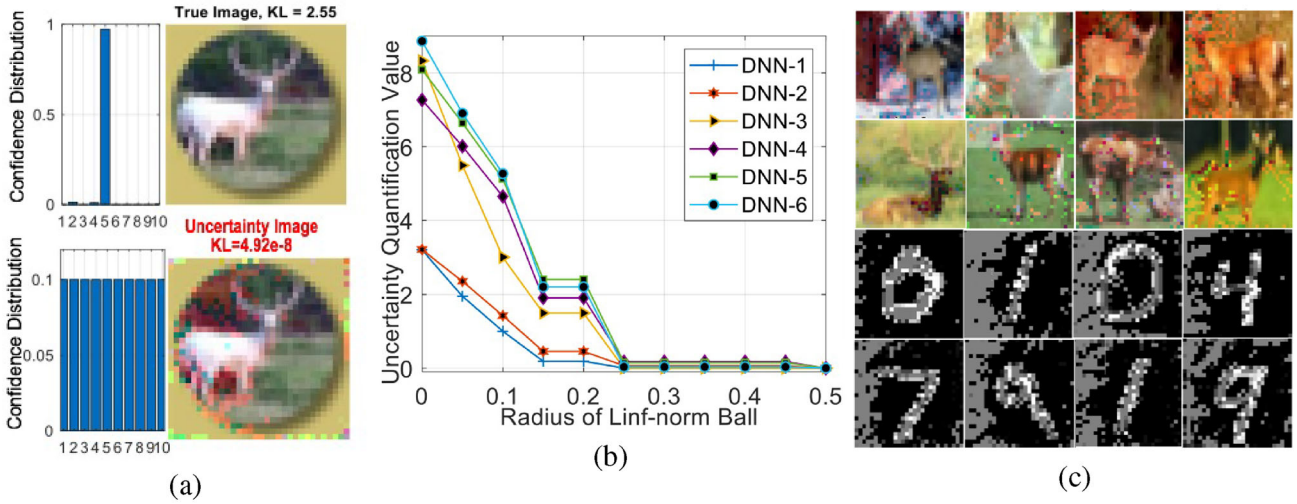


Fig. 10 **a** KL divergence between DNN's output distribution with uniform distribution for a true image and an uncertainty image. **b** Uncertainty values quantified by DEEPQUANT on six MNIST DNNs on a given L_∞-norm ball. **c** Uncertainty examples on MNIST and CIFAR-10 dataset in a L_∞-norm ball of radius $d = 0.4$

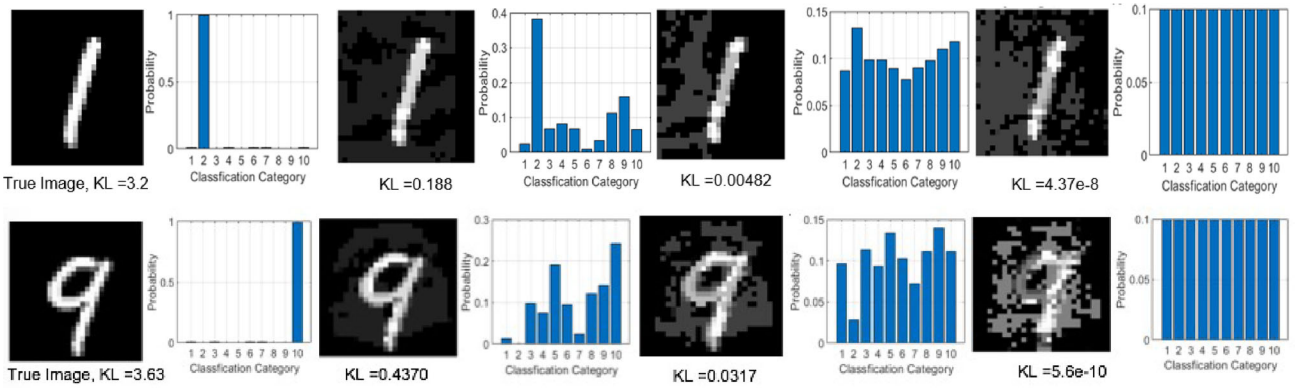


Fig. 11 Intermediate images obtained during the searching for an uncertainty image in a L_∞-norm ball with $d = 0.1$. From left to right, the KL divergence is gradually decreased

cannot work on large-scale neural networks. We only compare with random sampling (RS) method, and exhibit much more robustness values on MNIST and CIFAR-10 networks. More interesting, we explore the relationship between accuracy and robustness to design a robust models. As we present in Table 6, high accuracy is not corresponding to most robustness; hence, how to balance the accuracy and robustness is a critical problem of practical system for future work.

We first show the uncertainty examples and visualise the process for searching uncertainty images in this paper. We also obtain the uncertainty values using uncertainty expression. As our experiments show, the networks show vastly uncertain and confused to classify when gradually increasing norm-ball radius d from 0 to 0.5. Uncertainty reveals a new and important safety risks of neural network.

Game formalism of robustness verification The setting of the original adversarial attack [49] is actually a one-player game, with the attacker as the only player to optimise its objective. The adversarial training [30] can be seen as a concurrent game, where there are two players, training algorithm and attack algorithm, who know the existence of each other and act concurrently by considering the potential best response of each other. The concurrent game is hard to solve, and therefore, many implementations of the adversarial training are actually Stackelberg game, with one of the players as the "leader" to move first and the other as the "follower" to move after seeing the behaviour of the "leader". The robustness verification needs to consider the worst cases of whether or not the attacker can succeed, and therefore can work for all these settings.

Conclusion

In this paper, we present a novel method DeepQuant. Based on a generic Lipschitz metric and a derivative-free optimisation algorithm, DeepQuant can quantify a set of safety risks, including a new risk called uncertainty example. Different with adversarial examples, uncertainty examples provide a new insight and critical risk for human and deep learning systems when both of them are not confident with their own decision. Comparing with state-of-the-art methods, our method not only can work on a broad range of safety risks but also can return a tight result comparing to the ground truth. Our tool DeepQuant is optimized by tensor-based parallelisation, which could run efficiently on GPUs, and thus is scalable to work with large-scale networks including MNIST, CIFAR-10, and ImageNet models. We envision that this paper provides an initial yet important attempt towards the risk quantification concerning the safety of DNNs, and discuss the experimental results and some interesting direction in the future work. For example, the relationship between accuracy and robustness, especially in practical system, will attract much more attention.

Acknowledgements This work is supported by the UK EPSRC projects on Offshore Robotics for Certification of Assets (ORCA) [EP/R026173/1] and End-to-End Conceptual Guarding of Neural Architectures [EP/T026995/1], and ORCA Partnership Resource Fund (PRF) on Towards the Accountable and Explainable Learning-enabled Autonomous Robotic Systems [EP/R026173/1]. The corresponding author of this paper is Dr Wenjie Ruan.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical statements This article does not contain any studies involving animals performed by any of the authors. This article does not contain any studies involving human participants performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Anderson G, Pailoor S, Dillig I, Chaudhuri S (2019) Optimization and abstraction: a synergistic approach for analyzing neural network robustness. In: Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation, pp 731–744
2. Athalye A, Sutskever I (2018) Synthesizing robust adversarial examples. In: The 35th international conference on machine learning (ICML), pp 284–293
3. Audet C, Dennis JE (2000) Analysis of generalized pattern searches. SIAM J Optim 13:889–903
4. Audet C, Dennis JE Jr (2006) Mesh adaptive direct search algorithms for constrained optimization. SIAM J Optim 17(1):188–217
5. Audet C, Hare W (2017) Mesh adaptive direct search. In: Derivative-free and blackbox optimization. Springer, pp 135–156
6. Balan R, Singh M, Zou D (2017) Lipschitz properties for deep convolutional networks. arXiv:1701.05217
7. Bishop C (2006) Pattern recognition and machine learning. Springer, New York
8. Boopathy A, Weng TW, Chen PY, Liu S, Daniel L (2019) Cnn-cert: an efficient framework for certifying robustness of convolutional neural networks. Proc AAAI Conf Artif Intell 33:3240–3247
9. Bunel R, Turkaslan I, Torr PH, Kohli P, Kumar MP (2018) A unified view of piecewise linear neural network verification. In: Neural information processing systems (NIPS'18), pp 4790–4799
10. Carlini N, Wagner D (2017) Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on security and privacy (SP). IEEE, pp 39–57
11. Dutta S, Jha S, Sanakaranarayanan S, Tiwari A (2017) Output range analysis for deep neural networks. arXiv:1709.09130
12. Esteva A, Robicquet A, Ramsundar B, Kuleshov V, DePristo M, Chou K, Cui C, Corrado G, Thrun S, Dean J (2019) A guide to deep learning in healthcare. Nat Med 25(1):24–29
13. Galloway A, Taylor GW, Moussa M (2018) Attacking binarized neural networks. In: International conference on learning representations (ICLR)
14. Gehr T, Mirman M, Drachsler-Cohen D, Tsankov P, Chaudhuri S, Vechev M (2018) Ai2: safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE symposium on security and privacy (SP), pp 3–18
15. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge

16. Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and harnessing adversarial examples. In: International conference for learning representations (ICLR)
17. Hein M, Andriushchenko M (2017) Formal guarantees on the robustness of a classifier against adversarial manipulation. In: Neural information processing systems (NIPS), pp 2266–2276
18. Huang C, Hu Z, Huang X, Pei K (2021) Statistical certification of acceptable robustness for neural networks. In: Farkaš I, Masulli P, Otte S, Wermter S (eds) Artificial neural networks and machine learning—ICANN 2021. Springer International Publishing, Cham, pp 79–90
19. Huang W, Sun Y, Sharp J, Ruan W, Meng J, Huang X (2021) Coverage guided testing for recurrent neural networks. IEEE Trans Reliab 1–16
20. Huang X, Jin G, Ruan W (2022) Machine learning safety. Springer, Berlin
21. Huang X, Kroening D, Ruan W, Sharp J, Sun Y, Thamo E, Wu M, Yi X (2020) A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Comput Sci Rev 37:100270
22. Huang X, Kwiatkowska M, Wang S, Wu M (2017) Safety verification of deep neural networks. In: International conference on computer aided verification (CAV), pp 3–29
23. Ilyas A, Engstrom L, Athalye A, Lin J (2018) Black-box adversarial attacks with limited queries and information. In: The 35th international conference on machine learning (ICML), pp 2137–2146
24. Jacobsen JH, Behrmann J, Carlini N, Tramèr F, Papernot N (2020) Exploiting excessive invariance caused by norm-bounded adversarial robustness
25. Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ (2017) Reluplex: an efficient smt solver for verifying deep neural networks. In: International conference on computer aided verification, pp 97–117
26. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
27. Lewis RM, Torczon VJ, Kolda TG (2006) A generating set direct search augmented lagrangian algorithm for optimization with a combination of general and linear constraints. Tech. rep., Sandia National Laboratories
28. Li J, Liu J, Yang P, Chen L, Huang X, Zhang L (2019) Analyzing deep neural networks with symbolic propagation: towards higher precision and faster verification. In: Chang BYE (ed) Static analysis. Springer International Publishing, Cham, pp 296–319
29. Lomuscio A, Maganti L (2017) An approach to reachability analysis for feed-forward ReLU neural networks. arXiv:1706.07351
30. Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A (2018) Towards deep learning models resistant to adversarial attacks. In: International conference on learning representations. https://openreview.net/forum?id=rJzIBfZAb
31. Maqueda AI, Loquercio A, Gallego G, García N, Scaramuzza D (2018) Event-based vision meets deep learning on steering prediction for self-driving cars. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5419–5427
32. Mirman M, Gehr T, Vechev M (2018) Differentiable abstract interpretation for provably robust neural networks. In: International conference on machine learning (ICML), pp 3578–3586
33. Moosavi-Dezfooli SM, Fawzi A, Fawzi O, Frossard P (2017) Universal adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1765–1773
34. Mopuri KR, Ojha U, Garg U, Babu RV (2018) Nag: network for adversary generation. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 742–751
35. Mu R, Soriano Marcolino L, Ruan W, Ni Q (2021) Sparse adversarial video attacks with spatial transformations. In: 32nd British machine vision conference 2021, BMVC 2021
36. Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A (2016) The limitations of deep learning in adversarial settings. In: IEEE European symposium on security and privacy, pp 372–387
37. Peck J, Roels J, Goossens B, Saeys Y (2017) Lower bounds on the robustness to adversarial perturbations. In: Advances in neural information processing systems (NIPS), pp 804–813
38. Pérez-Cruz F (2009) Estimation of information theoretic measures for continuous random variables. In: Advances in neural information processing systems (NIPS), pp 1257–1264
39. Pulina L, Tacchella A (2010) An abstraction-refinement approach to verification of artificial neural networks. In: International conference on computer aided verification (CAV), pp 243–257
40. Raghunathan A, Steinhardt J, Liang PS (2018) Semidefinite relaxations for certifying robustness to adversarial examples. In: Neural information processing systems (NeurIPS), pp 10877–10887
41. Ruan W, Huang X, Kwiatkowska M (2018) Reachability analysis of deep neural networks with provable guarantees. In: International joint conference on artificial intelligence (IJCAI), pp 2651–2659
42. Ruan W, Wu M, Sun Y, Huang X, Kroening D, Kwiatkowska M (2019) Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence (IJCAI), pp 5944–5952
43. Ruan W, Yi X, Huang X (2021) Adversarial robustness of deep learning: Theory, algorithms, and applications. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp 4866–4869
44. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
45. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap T, Hui F, Sifre L, van den Driessche G, Graepel T, Hassabis D (2017) Mastering the game of Go without human knowledge. Nature 550:354–359
46. Su J, Vargas DV, Sakurai K (2019) One pixel attack for fooling deep neural networks. IEEE Trans Evol Comput 23(5):828–841
47. Sun Y, Huang X, Kroening D, Sharp J, Hill M, Ashmore R (2019) Structural test coverage criteria for deep neural networks. ACM Trans. Embed. Comput. Syst. 18(5s). https://doi.org/10.1145/3358233
48. Sun Y, Wu M, Ruan W, Huang X, Kwiatkowska M, Kroening D (2018) Concolic testing for deep neural networks. In: ASE2018
49. Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks. In: International conference on learning representations (ICLR)
50. Tjeng V, Xiao K, Tedrake R (2019) Evaluating robustness of neural networks with mixed integer programming. In: International conference on learning representations (ICLR)
51. Tramèr F, Kurakin A, Papernot N, Goodfellow I, Boneh D, McDaniel PD (2018) Ensemble adversarial training: attacks and defenses. In: International conference on learning representations (ICLR)
52. Tsipras D, Santurkar S, Engstrom L, Turner A, Madry A (2019) Robustness may be at odds with accuracy. In: International conference on learning representations (ICLR)
53. Webb S (2018) Deep learning for biology. Nature 554:7693
54. Weng TW, Zhang H, Chen H, Song Z, Hsieh CJ, Boning D, Dhillon IS, Daniel L (2018) Towards fast computation of certified robustness for relu networks. In: The 35th international conference on machine learning (ICML), pp 5276–5285
55. Wicker M, Huang X, Kwiatkowska M (2018) Feature-guided black-box safety testing of deep neural networks. In: International conference on tools and algorithms for the construction and analysis of systems (TACAS). Springer, pp 408–426

56. Wong E, Kolter JZ (2018) Provable defenses against adversarial examples via the convex outer adversarial polytope. In: International conference on machine learning (ICML), pp 5286–5295
57. Wu H, Ruan W (2021) Adversarial driving: attacking end-to-end autonomous driving systems. arXiv:2103.09151
58. Wu M, Wicker M, Ruan W, Huang X, Kwiatkowska M (2020) A game-based approximate verification of deep neural networks with provable guarantees. Theor Comput Sci 807:298–329
59. Xu K, Liu S, Zhao P, Chen PY, Zhang H, Fan Q, Erdogmus D, Wang Y, Lin X (2018) Structured adversarial attack: towards general implementation and better interpretability. In: International conference on learning representations (ICLR)
60. Xu P, Ruan W, Huang X (2020) Towards the quantification of safety risks in deep neural networks. arXiv:2009.06114
61. Zhang H, Weng TW, Chen PY, Hsieh CJ, Daniel L (2018) Efficient neural network robustness certification with general activation functions. Adv Neural Inf Process Syst 31:4939–4948
62. Zhang H, Zhang P, Hsieh CJ (2019) Recurjac: an efficient recursive algorithm for bounding Jacobian matrix of neural networks and its applications. Proc AAAI Conf Artif Intell 33:5757–5764
63. Zhang Y, Ruan W, Wang F, Huang X (2020) Generalizing universal adversarial attacks beyond additive perturbations. In: 2020 IEEE international conference on data mining (ICDM). IEEE, pp 1412–1417
64. Zhang Y, Wang F, Ruan W (2021) Fooling object detectors: adversarial attacks by half-neighbor masks. arXiv:2101.00989
65. Zhao X, Banks A, Sharp J, Robu V, Flynn D, Fisher M, Huang X (2020) A safety framework for critical systems utilising deep neural networks. In: SafeComp2020, pp 244–259
66. Zhao X, Huang W, Banks A, Cox V, Flynn D, Schewe S, Huang X (2021) Assessing reliability of deep learning through robustness evaluation and operational testing. In: AISafety2021
67. Zhao X, Huang W, Bharti V, Dong Y, Cox V, Banks A, Wang S, Schewe S, Huang X (2021) Reliability assessment and safety arguments for machine learning components in assuring learning-enabled autonomous systems. arXiv:2112.00646 [CoRR]

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.