A Game-Based Approximate Verification of Deep Neural Networks with Provable Guarantees

Min Wu[a], Matthew Wicker[b,1], Wenjie Ruan[a], Xiaowei Huang[c], Marta Kwiatkowska[a,*]

[a] *University of Oxford, UK*
[b] *University of Georgia, USA*
[c] *University of Liverpool, UK*

Abstract

Despite the improved accuracy of deep neural networks, the discovery of adversarial examples has raised serious safety concerns. In this paper, we study two variants of pointwise robustness, the *maximum safe radius* problem, which for a given input sample computes the minimum distance to an adversarial example, and the *feature robustness* problem, which aims to quantify the robustness of individual features to adversarial perturbations. We demonstrate that, under the assumption of Lipschitz continuity, both problems can be approximated using finite optimisation by discretising the input space, and the approximation has provable guarantees, i.e., the error is bounded. We then show that the resulting optimisation problems can be reduced to the solution of two-player turn-based games, where the first player selects features and the second perturbs the image within the feature. While the second player aims to minimise the distance to an adversarial example, depending on the optimisation objective the first player can be cooperative or competitive. We employ an anytime approach to solve the games, in the sense of approximating the value of a game by monotonically improving its upper and lower bounds. The Monte Carlo tree search algorithm is applied to compute upper bounds for both games, and the Admissible A* and the Alpha-Beta Pruning algorithms are, respectively, used to compute lower bounds for the maximum safety radius and feature robustness games. When working on the upper bound of the maximum safe radius problem, our tool demonstrates competitive performance against existing adversarial example crafting algorithms. Furthermore, we show how our framework can be deployed to evaluate pointwise robustness of neural networks in safety-critical applications such as traffic sign recognition in self-driving cars.

*Corresponding author

Email addresses: min.wu@cs.ox.ac.uk (Min Wu), matthew.wicker25@uga.edu (Matthew Wicker), wenjie.ruan@cs.ox.ac.uk (Wenjie Ruan), xiaowei.huang@liverpool.ac.uk (Xiaowei Huang), marta.kwiatkowska@cs.ox.ac.uk (Marta Kwiatkowska)

[1] Matthew conducted this research when he was a visiting undergraduate student at the University of Oxford.

Figure 1: An adversarial example for a neural network trained on the GTSRB dataset. After a slight perturbation of Euclidean distance 0.88, the image classification changes from "go **right** or straight" to "go **left** or straight".

Keywords: Automated Verification, Deep Neural Networks, Adversarial Examples, Two-Player Game

1. Introduction

Deep neural networks (DNNs or networks, for simplicity) have been developed for a variety of tasks, including malware detection [1], abnormal network activity detection [2], and self-driving cars [3, 4, 5]. A classification network N can be used as a decision-making algorithm: given an input α, it suggests a decision $N(\alpha)$ among a set of possible decisions. While the accuracy of neural networks has greatly improved, matching the cognitive ability of humans [6], they are susceptible to adversarial examples [7, 8]. An adversarial example is an input which, though initially classified correctly, is misclassified after a minor, perhaps imperceptible, perturbation. Adversarial examples pose challenges for self-driving cars, where neural network solutions have been proposed for tasks such as end-to-end steering [3], road segmentation [4], and traffic sign classification [5]. In the context of steering and road segmentation, an adversarial example may cause a car to steer off the road or drive into barriers, and misclassifying traffic signs may cause a vehicle to drive into oncoming traffic. Figure 1 shows an image of a traffic light correctly classified by a state-of-the-art network, which is then misclassified after only a few pixels have been changed. Though somewhat artificial, since in practice the controller would rely on additional sensor input when making a decision, such cases strongly suggest that, before deployment in safety-critical tasks, DNNs' resilience (or robustness) to adversarial examples must be strengthened.

Robustness of neural networks is an active topic of investigation and a number of approaches have been proposed to search for adversarial examples (see Related Work). They are based on computing the gradients [9], along which a heuristic search moves; computing a Jacobian-based saliency map [10], based on which pixels are selected to be changed; transforming the existence of adversarial examples into an optimisation problem [11], on which an optimisation algorithm can be applied; transforming the existence of adversarial examples into a constraint solving problem [12], on which a constraint solver can be applied; or discretising the neighbourhood of a point and searching it exhaustively in a layer-by-layer manner [13].

In this paper, we propose a novel game-based approach for safety verification of DNNs. We consider two pointwise robustness problems, referred to as the *maximum safe radius* problem and *feature robustness* problem, respectively. The former aims to compute for a given input the minimum distance to an adversarial example, and therefore can be regarded as the computation of an *absolute* safety radius, within which no adversarial example exists. The latter problem studies whether the crafting of adversarial examples can be controlled by restricting perturbations to only certain features (disjoint sets of input dimensions), and therefore can be seen as the computation of a *relative* safety radius, within which the existence of adversarial examples is controllable.

Both pointwise robustness problems are formally expressed in terms of non-linear optimisation, which is computationally challenging for realistically-sized networks. We thus utilise Lipschitz continuity of DNN layers, which bounds the maximal rate of change of outputs of a function with respect to the change of inputs, as proposed for neural networks with differentiable layers in [14, 15]. This enables safety verification by relying on Lipschitz constants to provide guaranteed bounds on DNN output *for all* possible inputs. We work with modern DNNs whose layers, e.g., ReLU, may not be differentiable, and reduce the verification to finite optimisation. More precisely, we prove that under the assumption of Lipschitz continuity [16] it is sufficient to consider a finite number of uniformly sampled inputs when the distances between the inputs are small, and that this reduction has provable guarantees, in the sense of the error being bounded by the distance between sampled inputs.

We then show that the finite optimisation problems can be computed as the solution of two-player turn-based games, where Player `I` selects features and Player `II` then performs a perturbation within the selected features. After both players have made their choices, the input is perturbed and the game continues. While Player `II` aims to minimise the distance to an adversarial example, Player `I` can be *cooperative* or *competitive*. When it is cooperative, the optimal reward of Player `I` is equal to the maximum safe radius. On the other hand, when it is competitive the optimal reward of Player `I` quantifies feature robustness. Finally, because the state space of the game models is intractable, we employ an anytime approach to compute the upper and lower bounds of Player `I` optimal reward. The anytime approach ensures that the bounds can be gradually, but strictly, improved so that they eventually converge. More specifically, we apply Monte Carlo tree search algorithm to compute the upper bounds for both games, and Admissible A* and Alpha-Beta Pruning, respectively, to compute the lower bounds for the games.

We implement the method in a software tool DeepGame[2], and conduct experiments on DNNs to show convergence of lower and upper bounds for the maximum safe radius and feature robustness problems. Our approach can be configured to work with a variety of feature extraction methods that partition the input, for example image segmentation, with simple adaptations. For

[2]The software package is available from `https://github.com/TrustAI/DeepGame`

the image classification networks we consider in the experiments, we employ both the saliency-guided grey-box approach adapted from [17] and the feature-guided black-box method based on the SIFT object detection technique [18]. For the maximum safety radius problem, our experiments show that, on networks trained on the benchmark datasets such as MNIST [19], CIFAR10 [20] and GTSRB [21], the upper bound computation method is competitive with state-of-the-art heuristic methods (i.e., without provable guarantees) that rely on white-box saliency matrices or sophisticated optimisation procedures. Finally, to show that our framework is well suited to safety testing and decision support for deploying DNNs in safety-critical applications, we experiment on state-of-the-art networks, including the winner of the Nexar traffic light challenge [22].

The paper significantly extends work published in [23], where the game-based approach was first introduced for the case of cooperative games and evaluated on the computation of upper bounds for the maximum safety radius problem using the SIFT feature extraction method. In contrast, in this paper we additionally study feature robustness, generalise the game to allow for the competitive player, and develop algorithms for the computation of both lower and upper bounds. We also give detailed proofs of the theoretical guarantees and error bounds.

The structure of the paper is as follows. After introducing preliminaries in Section 2, we formalise the maximum safety radius and feature robustness problems in Section 3. We present our game-based approximate verification approach and state the guarantees in Section 4. Algorithms and implementation are described in Section 5, while experimental results are given in Section 6. We discuss the related work in Section 7 and conclude the paper in Section 8.

2. Preliminaries

Let N be a neural network with a set C of classes. Given an input α and a class $c \in C$, we use $N(\alpha, c)$ to denote the confidence (expressed as a probability value obtained from normalising the score) of N believing that α is in class c. Moreover, we write $N(\alpha) = \arg\max_{c \in C} N(\alpha, c)$ for the class into which N classifies α. We let P_0 be the set of input dimensions, $n = |P_0|$ be the number of input dimensions, and remark that without loss of generality the dimensions of an input are normalised as real values in $[0, 1]$. The input domain is thus a vector space

$$\mathrm{D} = [0, 1]^n.$$

For image classification networks, the input domain D can be represented as $[0, 1]^{w \times h \times ch}_{[0,255]}$, where w, h, ch are the width, height, and number of channels of an image, respectively. That is, we have $P_0 = w \times h \times ch$. We may refer to an element in $w \times h$ as a *pixel* and an element in P_0 as a *dimension*. We use $\alpha[i]$ for $i \in P_0$ to denote the value of the i-th dimension of α.

2.1. Distance Metric and Lipschitz Continuity

As is common in the field, we will work with L_k distance functions to measure the distance between inputs, denoted $||\alpha - \alpha'||_k$ with $k \geq 1$, and satisfying the standard axioms of a metric space:

- $||\alpha - \alpha'||_k \geq 0$ (non-negativity),
- $||\alpha - \alpha'||_k = 0$ implies that $\alpha = \alpha'$ (identity of indiscernibles),
- $||\alpha - \alpha'||_k = ||\alpha' - \alpha||_k$ (symmetry),
- $||\alpha - \alpha''||_k \leq ||\alpha - \alpha'||_k + ||\alpha' - \alpha''||_k$ (triangle inequality).

While we focus on L_k distances, including L_1 (Manhattan distance), L_2 (Euclidean distance), and L_∞ (Chebyshev distance), we emphasise that the results of this paper hold for any distance metric and can be adapted to image similarity distances such as SSIM [24]. Though our results do not generalise to L_0 (Hamming distance), we utilise it for the comparison with existing approaches to generate adversarial examples, i.e., without provable guarantees (Section 6.4).

Since we work with pointwise robustness [25], we need to consider the *neighbourhood* of a given input.

Definition 1. *Given an input α, a distance function L_k, and a distance d, we define the* d-neighbourhood $\eta(\alpha, L_k, d)$ *of α wrt L_k*

$$\eta(\alpha, L_k, d) = \{\alpha' \mid ||\alpha' - \alpha||_k \leq d\}$$

as the set of inputs whose distance to α is no greater than d with respect to L_k.

The d-neighbourhood of α is simply the L_k ball with radius d. For example, $\eta(\alpha, L_1, d)$ includes those inputs such that the sum of the differences of individual dimensions from the original input α is no greater than d, i.e., $||\alpha' - \alpha||_1 = \sum_{i \in P_0} |\alpha[i] - \alpha'[i]|$. Furthermore, we have $||\alpha' - \alpha||_2 = \sqrt{\sum_{i \in P_0} (\alpha[i] - \alpha'[i])^2}$ and $||\alpha' - \alpha||_\infty = \max_{i \in P_0} |\alpha[i] - \alpha'[i]|$. We will sometimes work with d^ϵ-neighbourhood, where, given a number d, $d^\epsilon = d + \epsilon$ for any real number $\epsilon > 0$ denotes a number greater than d.

We will restrict the neural networks we consider to those that satisfy the *Lipschitz continuity* assumption, noting that all networks whose inputs are bounded, including all image classification networks we studied, are Lipschitz continuous. Specifically, it is shown in [25, 16] that most known types of layers, including fully-connected, convolutional, ReLU, maxpooling, sigmoid, softmax, etc., are Lipschitz continuous.

Definition 2. *Network N is a Lipschitz network with respect to distance function L_k if there exists a constant $\hbar_c > 0$ for every class $c \in C$ such that, for all $\alpha, \alpha' \in \mathrm{D}$, we have*

$$|N(\alpha', c) - N(\alpha, c)| \leq \hbar_c \cdot ||\alpha' - \alpha||_k, \tag{1}$$

where $\hbar_c$ is the Lipschitz constant *for class c.*

2.2. Input Manipulations

To study the crafting of adversarial examples, we require the following operations for manipulating inputs. Let $\tau > 0$ be a positive real number representing the manipulation magnitude, then we can define *input manipulation* operations $\delta_{\tau,X,\psi} : \mathrm{D} \to \mathrm{D}$ for $X \subseteq P_0$, a subset of input dimensions, and $\psi : P_0 \to \mathbb{N}$, an instruction function by:

$$\delta_{\tau,X,\psi}(\alpha[i]) = \begin{cases} \alpha[i] + \psi(i) * \tau, & \text{if } i \in X \\ \alpha[i], & \text{otherwise} \end{cases} \tag{2}$$

for all $i \in P_0$. Note that if the values are bounded, e.g., in the interval $[0, 1]$, then $\delta_{\tau,X,\psi}(\alpha[i])$ needs to be restricted to be within the bounds. Let Ψ be the set of possible instruction functions.

The following lemma shows that input manipulation operations allow one to map one input to another by changing the values of input dimensions, regardless of the distance measure L_k.

Lemma 1. *Given any two inputs α_1 and α_2, and a distance $||\alpha_1 - \alpha_2||_k$ for any measure L_k, there exists a magnitude $\tau > 0$, an instruction function $\psi \in \Psi$, and a subset $X \subseteq P_0$ of input dimensions, such that*

$$||\alpha_2 - \delta_{\tau,X,\psi}(\alpha_1)||_k \leq \epsilon$$

where $\epsilon > 0$ is an error bound.

Intuitively, any distance can be implemented through an input manipulation with an error bound ϵ. The error bound ϵ is needed because input $\alpha \in \mathrm{D} = [0, 1]^n$ is bounded, and thus reaching another precise input point via a manipulation is difficult when each input dimension is a real number.

We will also distinguish a subset of *atomic* input manipulations, each of which changes a single dimension for a single magnitude.

Definition 3. *Given a set X, we let $\Delta(X)$ be the set of* atomic *input manipulations δ_{τ,X_1,ψ_1} such that*

- *$X_1 \subseteq X$ and $|X_1| = 1$, and*
- *$\psi_1(i) \in \{-1, +1\}$ for all $i \in P_0$.*

Lemma 2. *Any input manipulation $\delta_{\tau,X,\psi}(\alpha)$ for some X and ψ can be implemented with a finite sequence of input manipulations $\delta_{\tau,X_1,\psi_1}(\alpha), ..., \delta_{\tau,X_m,\psi_m}(\alpha) \in \Delta(X)$.*

While the existence of a sequence of atomic manipulations implementing a given manipulation is determined, there may exist multiple sequences. On the other hand, from a given sequence of atomic manipulations we can construct a single input manipulation by sequentially applying the atomic manipulations.

2.3. Feature-Based Partitioning

Natural data, for example natural images and sounds, forms a high-dimensional manifold, which embeds tangled manifolds to represent their features [26]. Feature manifolds usually have lower dimensions than the data manifold. Intuitively, the set of features form a partition of the input dimensions P_0. In this paper, we use a variety of feature extraction methods to partition the set P_0 into disjoint subsets.

Definition 4. *Let λ be a feature of an input $\alpha \in \mathrm{D}$, then we use $P_\lambda \subseteq P_0$ to denote the dimensions represented by λ. Given an input α, a* feature extraction *function Λ maps an input α into a set of features $\Lambda(\alpha)$ such that (1) $P_0 = \bigcup_{\lambda \in \Lambda(\alpha)} P_\lambda$, and (2) $P_{\lambda_i} \cap P_{\lambda_j} = \emptyset$ for any $\lambda_i, \lambda_j \in \Lambda(\alpha)$ with $i \neq j$.*

We remark that our technique is not limited to image classification networks and is able to work with general classification tasks, as long as there is a suitable feature extraction method that generates a partition of the input dimensions. In our experiments we focus on image classification for illustrative purposes and to enable better comparison, and employ *saliency-guided* grey-box and *feature-guided* black-box approaches to extract features, described in Section 4.

3. Problem Statement

In this paper we focus on *pointwise robustness* [25], defined as the invariance of the network's classification over a small neighbourhood of a given input. This is a key concept, which also allows one to define robustness as a network property, by averaging with respect to the distribution of the test data set. Pointwise robustness can be used to define *safety* of a classification decision for a specific input, understood as the non-existence of an adversarial example in a small neighbourhood of the input. We work with this notion and consider two problems for quantifying the robustness of the decision, the computation of the *maximum safe radius* and *feature robustness*, which we introduce next.

First we recall the concept of an *adversarial example*, as well as what we mean by *targeted* and *non-targeted safety*.

Definition 5. *Given an input $\alpha \in \mathrm{D}$, a distance measure L_k for some $k \geq 0$, and a distance d, an* adversarial example *α' of class c is such that $\alpha' \in \eta(\alpha, L_k, d)$, $N(\alpha) \neq N(\alpha')$, and $N(\alpha') = c$. Moreover, we write $adv_{k,d}(\alpha, c)$ for the set of adversarial examples of class c and let*

$$adv_{k,d}(\alpha) = \bigcup_{c \in C, c \neq N(\alpha)} adv_{k,d}(\alpha, c).$$

A targeted safety *of class c is defined as $adv_{k,d}(\alpha, c) = \emptyset$, and* non-targeted safety *is $adv_{k,d}(\alpha) = \emptyset$.*

The following formalisation focuses on targeted safety of a fixed input α and a fixed class $c \neq N(\alpha)$ for a network N. The case of non-targeted safety (misclassification into class other than c) is similar.

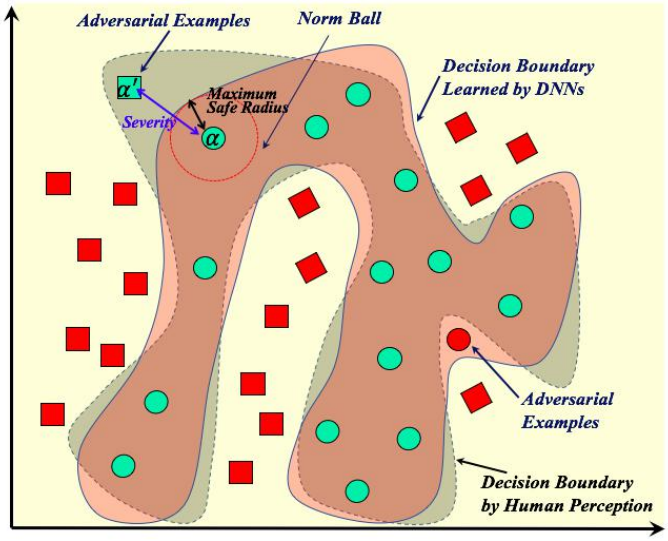


Figure 2: The *Maximum Safe Radius* (MSR) problem aims to quantify the minimum distance from an original image α to an adversarial example, equivalent to finding the radius of a maximum safe norm ball. The solid line represents the classification boundary learned by a DNN, while the dashed line is the decision boundary. Adversarial examples tend to lie where the decision and classification boundaries do not align. Intuitively, finding an adversarial example (green square) can only provide a loose upper bound of MSR.

3.1. The Maximum Safe Radius Problem

Given a targeted safety problem for α, we aim to compute the distance $||\alpha - \alpha'||_k$ to the nearest adversarial example within the d-neighbourhood of α, or in other words the radius of the maximum safe ball, illustrated in Figure 2.

Definition 6 (Maximum Safe Radius). *The* maximum safe radius *problem is to compute the minimum distance from the original input α to an adversarial example, i.e.,*

$$\texttt{MSR}(k, d, \alpha, c) = \min_{\alpha' \in \mathrm{D}} \{||\alpha - \alpha'||_k \mid \alpha' \in adv_{k,d}(\alpha, c)\} \tag{3}$$

If $adv_{k,d}(\alpha, c) = \emptyset$, we let $\texttt{MSR}(k, d, \alpha, c) = d^{\epsilon}$.

Intuitively, $\texttt{MSR}(k, d, \alpha, c)$ represents an *absolute* safety radius within which all inputs are safe. In other words, within a distance of less than $\texttt{MSR}(k, d, \alpha, c)$, no adversarial example is possible. When no adversarial example can be found within radius d, i.e., $adv_{k,d}(\alpha, c) = \emptyset$, the maximum safe radius cannot be computed, but is definitely greater than d. Therefore, we let $\texttt{MSR}(k, d, \alpha, c) = d^{\epsilon}$.

Intuitively, finding an adversarial example can only provide a loose upper bound of MSR. Instead, this paper investigates a more fundamental problem – how to approximate the *true* MSR distance with provable guarantees.

Approximation Based on Finite Optimisation. Note that the sets $adv_{k,d}(\alpha, c)$ and $adv_{k,d}(\alpha)$ of adversarial examples can be infinite. We now present a discretisation method that allows us to approximate the maximum safe radius using finite optimisation, and show that such a reduction has provable guarantees, provided that the network is Lipschitz continuous. Our approach proceeds by constructing a finite 'grid' of points in the input space. Lipschitz continuity enables us to reduce the verification problem to manipulating just the grid points, through which we can bound the output behaviour of a DNN on the whole input space, since Lipschitz continuity ensures that the network behaves well within each cell. The number of grid points is inversely proportional to the Lipschitz constant. However, estimating a tight Lipschitz constant is difficult, and so, rather than working with the Lipschitz constant directly, we assume the existence of a (not necessarily tight) Lipschitz constant and work instead with a chosen fixed magnitude of an input manipulation, $\tau \in (0, 1]$. We show how to determine the largest τ for a given Lipschitz network and give error bounds for the computation of $\mathtt{MSR}$ that depend on τ. We discuss how Lipschitz constants can be estimated in Section 6.2 and Related Work.

We begin by constructing, for a chosen fixed magnitude $\tau \in (0, 1]$, input manipulations to search for adversarial examples.

Definition 7. *Let $\tau \in (0, 1]$ be a manipulation magnitude. The* finite maximum safe radius *problem $\mathtt{FMSR}(\tau, k, d, \alpha, c)$ based on input manipulation is as follows:*

$$\min_{\Lambda' \subseteq \Lambda(\alpha)} \min_{X \subseteq \bigcup_{\lambda \in \Lambda'} P_\lambda} \min_{\psi \in \Psi} \{||\alpha - \delta_{\tau,X,\psi}(\alpha)||_k \mid \delta_{\tau,X,\psi}(\alpha) \in adv_{k,d}(\alpha, c)\}. \quad (4)$$

If $adv_{k,d}(\alpha, c) = \emptyset$, we let $\mathtt{FMSR}(\tau, k, d, \alpha, c) = d^\epsilon$.

Intuitively, we aim to find a set Λ' of features, a set X of dimensions within Λ', and a manipulation instruction ψ such that the application of the atomic manipulation $\delta_{\tau,X,\psi}$ on the original input α leads to an adversarial example $\delta_{\tau,X,\psi}(\alpha)$ that is nearest to α among all adversarial examples. Compared to Definition 6, the search for another input by $\min_{\alpha' \in \mathrm{D}}$ over an infinite set is implemented by minimisation over the finite sets of feature sets and instructions.

Since the set of input manipulations is finite for a fixed magnitude, the above optimisation problems need only explore a finite number of 'grid' points in the input domain D. We have the following lemma.

Lemma 3. *For any $\tau \in (0, 1]$, we have that $\mathtt{MSR}(k, d, \alpha, c) \leq \mathtt{FMSR}(\tau, k, d, \alpha, c)$.*

To ensure the lower bound of $\mathtt{MSR}(k, d, \alpha, c)$ in Lemma 3, we utilise the fact that the network is Lipschitz continuous [16]. First, we need the concepts of a τ-grid input, for a manipulation magnitude τ, and a misclassification aggregator. The intuition for the τ-grid is illustrated in Figure 3. We construct a finite set of grid points uniformly spaced by τ in such a way that they can be covered by small subspaces centred on grid points. We select a sufficiently small value for τ based on a given Lipschiz constant so that all points in these subspaces

are are classified the same. We then show that an optimum point on the grid is within an error bound dependent on τ from the true optimum, i.e., the closest adversarial example.

Definition 8. *An image $\alpha' \in \eta(\alpha, L_k, d)$ is a* τ-grid input *if for all dimensions $p \in P_0$ we have $|\alpha'(p) - \alpha(p)| = n * \tau$ for some $n \geq 0$. Let $G(\alpha, k, d)$ be the set of τ-grid inputs in $\eta(\alpha, L_k, d)$.*

We note that τ-grid inputs in the set $G(\alpha, k, d)$ are reachable from each other by applying an input manipulation. The main purpose of defining τ-grid inputs is to ensure that the space $\eta(\alpha, L_k, d)$ can be covered by small subspaces centred on grid points. To implement this, we need the following lemma.

Lemma 4. *We have $\eta(\alpha, L_k, d) \subseteq \bigcup_{\alpha' \in G(\alpha,k,d)} \eta(\alpha', L_k, \frac{1}{2}d(k, \tau))$, where $d(k, \tau) = (|P_0|\tau^k)^{\frac{1}{k}}$.*

Proof: Let α_1 be any point in $\eta(\alpha, L_k, d)$. We need to show $\alpha_1 \in \eta(\alpha', L_k, \frac{1}{2}d(k, \tau))$ for some τ-grid input α'. Because every point in $\eta(\alpha, L_k, d)$ belongs to a τ-grid cell, we assume that α_1 is in a τ-grid cell which, without loss of generality, has a set T of τ-grid inputs as its vertices. Now for any two τ-grid inputs α_2 and α_3 in T, we have that $||\alpha_2 - \alpha_3||_k \leq d(k, \tau)$, by the construction of the grid. Therefore, we have $\alpha_1 \in \eta(\alpha', L_k, \frac{1}{2}d(k, \tau))$ for some $\alpha' \in T$. □

As shown in Figure 3, the distance $\frac{1}{2}d(k, \tau)$ is the radius of norm ball subspaces covering the input space. It is easy to see that $d(1, \tau) = |P_0|\tau$, $d(2, \tau) = \sqrt{|P_0|\tau^2}$, and $d(\infty, \tau) = \tau$.

Definition 9. *An input $\alpha_1 \in \eta(\alpha, L_k, d)$ is a* misclassification aggregator *with respect to a number $\beta > 0$ if, for any $\alpha_2 \in \eta(\alpha_1, L_k, \beta)$, we have that $N(\alpha_2) \neq N(\alpha)$ implies $N(\alpha_1) \neq N(\alpha)$.*

Intuitively, if a misclassification aggregator α_1 with respect to β is classified correctly, then *all* inputs in $\eta(\alpha_1, L_k, \beta)$ are classified correctly.

Error Bounds. We now bound the error of using $\texttt{FMSR}(\tau, k, d, \alpha, c)$ to estimate $\texttt{MSR}(k, d, \alpha, c)$ in $\frac{1}{2}d(k, \tau)$, as illustrated in Figure 3. First of all, we have the following lemma. Recall from Lemma 3 that we already have $\texttt{MSR}(k, d, \alpha, c) \leq \texttt{FMSR}(\tau, k, d, \alpha, c)$.

Lemma 5. *If all τ-grid inputs are misclassification aggregators with respect to $\frac{1}{2}d(k, \tau)$, then $\texttt{MSR}(k, d, \alpha, c) \geq \texttt{FMSR}(\tau, k, d, \alpha, c) - \frac{1}{2}d(k, \tau)$.*

Proof: We prove by contradiction. Assume that $\texttt{FMSR}(\tau, k, d, \alpha, c) = d'$ for some $d' > 0$, and $\texttt{MSR}(k, d, \alpha, c) < d' - \frac{1}{2}d(k, \tau)$. Then there must exist an input α' such that $\alpha' \in adv_{k,d}(\alpha, c)$ and

$$||\alpha' - \alpha||_k = \texttt{MSR}(k, d, \alpha, c) < d' - \frac{1}{2}d(k, \tau), \tag{5}$$

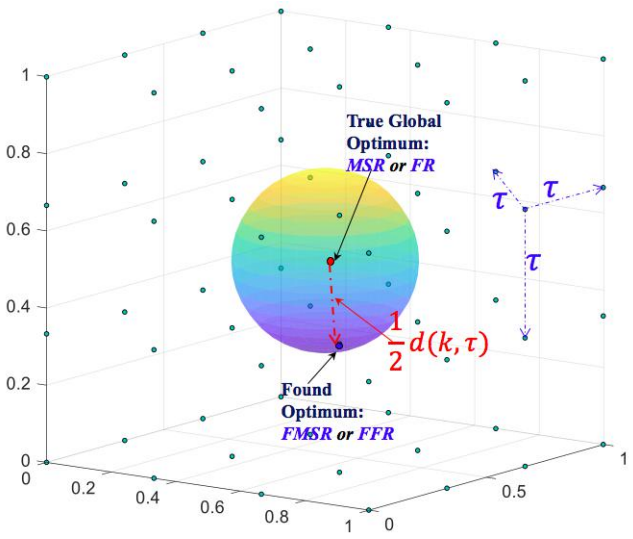


Figure 3: Provable guarantees for the MSR and $\texttt{FR}_\Lambda$ problems on a dense τ-grid (green dots) that is reached upon convergence. In the worst case, the true optimum (the red dot) lies in the middle between two hyper-points of the τ-grid, with the distance of at most $\frac{1}{2}d(k,\tau)$ from the found optimum.

and α' is not a τ-grid input. By Lemma 4, there must exist a τ-grid input α'' such that $\alpha' \in \eta(\alpha'', L_k, \frac{1}{2}d(k,\tau))$. Now because all τ-grid inputs are misclassification aggregators with respect to $\frac{1}{2}d(k,\tau)$, we have $\alpha'' \in adv_{k,d}(\alpha, c)$.

By $\alpha'' \in adv_{k,d}(\alpha, c)$ and the fact that α'' is a τ-grid input, we have that

$$\texttt{FMSR}(\tau, k, d, \alpha, c) \leq ||\alpha - \alpha''||_k \leq ||\alpha - \alpha'||_k + \frac{1}{2}d(k,\tau). \tag{6}$$

Now, combining Equations (5) and (6), we have that $\texttt{FMSR}(\tau, k, d, \alpha, c) < d'$, which contradicts the hypothesis that $\texttt{FMSR}(\tau, k, d, \alpha, c) = d'$. □

In the following, we discuss how to determine the largest τ for a Lipschitz network in order to satisfy the condition in Lemma 5 that all τ-grid inputs are misclassification aggregators with respect to $\frac{1}{2}d(k,\tau)$.

Definition 10. *Given a class label c, we let*

$$g(\alpha', c) = \min_{c' \in C, c' \neq c} \{N(\alpha', c) - N(\alpha', c')\} \tag{7}$$

be a function maintaining for an input α' the minimum confidence margin *between the class c and another class $c' \neq N(\alpha')$.*

Note that, given an input α' and a class c, we can compute $g(\alpha', c)$ in constant time. The following lemma shows that the above-mentioned condition about misclassification aggregators can be obtained if τ is sufficiently small.

Lemma 6. *Let N be a Lipschitz network with a Lipschitz constant $\hbar_c$ for every class $c \in C$. If*

$$d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c\in C, c\neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_c)} \tag{8}$$

for all τ-grid input $\alpha' \in G(\alpha, k, d)$, then all τ-grid inputs are misclassification aggregators with respect to $\frac{1}{2}d(k,\tau)$.

Proof: For any input α'' whose closest τ-grid input is α', we have

$$\begin{aligned}
&g(\alpha', N(\alpha')) - g(\alpha'', N(\alpha')) \\
&= \min_{c\in C, c\neq N(\alpha')}\{N(\alpha', N(\alpha')) - N(\alpha', c)\} - \min_{c'\in C, c'\neq N(\alpha')}\{N(\alpha'', N(\alpha')) - N(\alpha'', c')\} \\
&\leq \max_{c'\in C, c'\neq N(\alpha')}\{N(\alpha', N(\alpha')) - N(\alpha', c') - N(\alpha'', N(\alpha')) + N(\alpha'', c')\} \\
&\leq \max_{c'\in C, c'\neq N(\alpha')}\{|N(\alpha', N(\alpha')) - N(\alpha'', N(\alpha'))| + |N(\alpha'', c') - N(\alpha', c')|\} \\
&\leq \max_{c'\in C, c'\neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})||\alpha' - \alpha''||_k \\
&\leq \max_{c'\in C, c'\neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})\frac{1}{2}d(k,\tau)
\end{aligned} \tag{9}$$

Now, to ensure that no class change occurs between α'' and α', we need to have $g(\alpha'', N(\alpha')) \geq 0$, which means that $g(\alpha', N(\alpha')) - g(\alpha'', N(\alpha')) \leq g(\alpha', N(\alpha'))$. Therefore, we can let

$$\max_{c'\in C, c'\neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})\frac{1}{2}d(k,\tau) \leq g(\alpha', N(\alpha')). \tag{10}$$

Note that $g(\alpha', N(\alpha'))$ is dependent on the τ-grid input α', and thus can be computed when we construct the grid. Finally, we let

$$d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c'\in C, c'\neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})} \tag{11}$$

Therefore, if we have the above inequality for every τ-grid input, then we can conclude $g(\alpha'', N(\alpha')) \geq 0$ for any $\alpha'' \in \eta(\alpha', k, d)$, i.e., $N(\alpha'', N(\alpha')) \geq N(\alpha'', c)$ for all $c \in C$. The latter means that no class change occurs. □

Combining Lemmas 3, 5, and 6, we have the following theorem which shows that the reduction has a provable guarantee, dependent on the choice of the manipulation magnitude.

Theorem 1. *Let N be a Lipschitz network with a Lipschitz constant $\hbar_c$ for every class $c \in C$. If $d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c'\in C, c'\neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})}$ for all τ-grid inputs $\alpha' \in G(\alpha, k, d)$, then we can use $\texttt{FMSR}(\tau, k, d, \alpha, c)$ to estimate $\texttt{MSR}(k, d, \alpha, c)$ with an error bound $\frac{1}{2}d(k,\tau)$.*

Proof: By Lemma 3, we have $\texttt{MSR}(k, d, \alpha, c) \leq \texttt{FMSR}(\tau, k, d, \alpha, c)$ for any $\tau > 0$. By Lemma 5 and Lemma 6, when $\texttt{FMSR}(\tau, k, d, \alpha, c) = d'$, we have $\texttt{MSR}(k, d, \alpha, c) \geq$

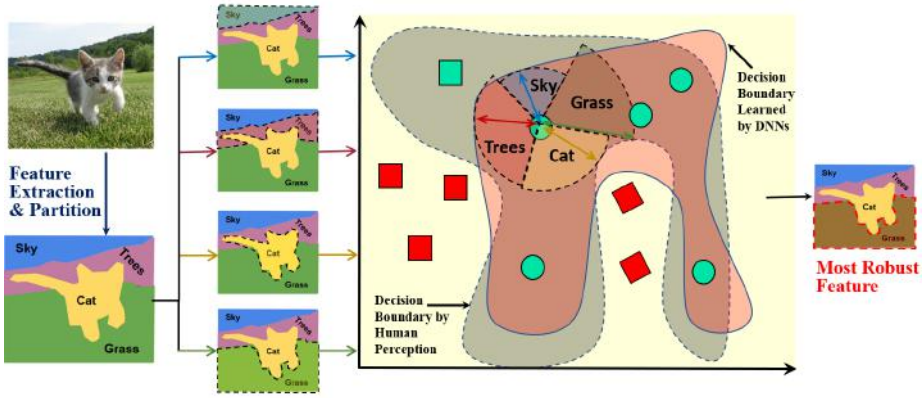


Figure 4: Illustration of the *feature robustness* ($\texttt{FR}_\Lambda$) problem, which aims to find, on an original image α, a feature, or a subset of features, that is the most robust against adversarial perturbations. Given a benign image, we first apply feature extraction or semantic partitioning methods to produce a set of disjoint features ('Sky', 'Trees', 'Cat', etc.), then we find a set of robust features that is most resilient to adversarial perturbations ('Grass' in the figure), which quantifies the most robust direction in a safe norm ball.

$d' - \frac{1}{2}d(k,\tau)$, under the condition that $d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})}$ for all τ-grid inputs $\alpha' \in G(\alpha, k, d)$. Therefore, when $d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})}$ for all τ-grid inputs $\alpha' \in G(\alpha, k, d)$, if we use d' to estimate $\texttt{MSR}(k, d, \alpha, c)$, we will have $d' - \frac{1}{2}d(k,\tau) \leq \texttt{MSR}(k, d, \alpha, c) \leq d'$, i.e., the error bound is $\frac{1}{2}d(k,\tau)$. □

3.2. The Feature Robustness Problem

The second problem studied in this paper concerns which features are the most robust against perturbations, illustrated in Figure 4. The *feature robustness* problem has been studied in explainable AI. For example, [27] explains the decision making of an image classification network through different contributions of the superpixels (i.e., features) and [28] presents a general additive model for explaining the decisions of a network by the Shapley value computed over the set of features.

Let $P_0(\alpha_1, \alpha_2) \subseteq P_0$ be the set of input dimensions on which α_1 and α_2 have different values.

Definition 11 (Feature Robustness). *The* feature robustness *problem is defined as follows.*

$$\texttt{FR}_\Lambda(k, d, \alpha, c) = \max_{\lambda \in \Lambda(\alpha)} \{\texttt{xFR}_\Lambda(\lambda, k, d, \alpha, c)\} \tag{12}$$

where $\texttt{xFR}_\Lambda(\lambda_m, k, d, \alpha_m, c) =$

$$\begin{cases} \min\limits_{\alpha_{m+1} \in \eta(\alpha, L_k, d)} \{||\alpha_m - \alpha_{m+1}||_k + \texttt{FR}_\Lambda(k, d, \alpha_{m+1}, c) \mid \\ \qquad\qquad \emptyset \neq P_0(\alpha_m, \alpha_{m+1}) \subseteq P_{\lambda_m}\}, & \text{if } \alpha_m \notin adv_{k,d}(\alpha, c) \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

where Λ *is a feature extraction function, and* $\alpha_m, \alpha_{m+1} (m \in \mathbb{N})$ *are the inputs before and after the application of some manipulation on a feature* λ_m*, respectively. If after selecting a feature* λ_m *no adversarial example can be reached, i.e.,* $\forall \alpha_{m+1} : P_0(\alpha_m, \alpha_{m+1}) \subseteq P_{\lambda_m} \Rightarrow \alpha_{m+1} \notin adv_{k,d}(\alpha, c)$*, then we let* $\texttt{xFR}_\Lambda(\lambda_m, k, d, \alpha_m, c) = d^\epsilon$*.*

Intuitively, the search for the most robust feature alternates between maximising over the features and minimising over the possible input dimensions within the selected feature, with the distance to the adversarial example as the objective. Starting from $\texttt{FR}_\Lambda(k, d, \alpha_0, c)$ where α_0 is the original image, the process moves to $\texttt{FR}_\Lambda(k, d, \alpha_1, c)$ by a max-min alternation on selecting feature λ_0 and next input α_1. This continues until either an adversarial example is found, or the next input α_i for some $i > 0$ is outside the d-neighbourhood $\eta(\alpha, L_k, d)$. The value d^ϵ is used is to differentiate from the case where the minimal adversarial example has exactly distance d from α_0 and the manipulations are within λ_0. In such a case, according to Equation (13), we have $\texttt{xFR}_\Lambda(\lambda_0, k, d, \alpha_0, c) = d$.

Assuming $\texttt{FR}_\Lambda(k, d, \alpha, c)$ has been computed and a *distance budget* $d' \leq d$ is given to manipulate the input α, the following cases can be considered.

- If $\texttt{FR}_\Lambda(k, d, \alpha, c) > d$, then there are robust features, and if manipulations are restricted to those features no adversarial example is possible.
- If $\texttt{FR}_\Lambda(k, d, \alpha, c) \leq d' \leq d$, then, no matter how one restricts the features to be manipulated, an adversarial example can be found within the budget.
- If $\texttt{MSR}(k, d, \alpha, c) \leq d' < \texttt{FR}_\Lambda(k, d, \alpha, c) \leq d$, then the existence of adversarial examples is controllable, i.e., we can choose a set of features on which the given distance budget d' is insufficient to find an adversarial example. This differs from the first case in that an adversarial example can be found if given a larger budget d.

Therefore, studying the feature robustness problem enables a better understanding of the robustness of individual features and how the features contribute to the robustness of an image.

It is straightforward to show that

$$\texttt{MSR}(k, d, \alpha, c) \leq \texttt{FR}_\Lambda(k, d, \alpha, c). \tag{14}$$

Compared to the absolute safety radius by $\texttt{MSR}(k, d, \alpha, c)$, $\texttt{FR}_\Lambda(k, d, \alpha, c)$ can be seen as a *relative* safety radius, within which the existence of adversarial examples can be controlled. Theoretically, the $\texttt{MSR}(k, d, \alpha, c)$ problem can be

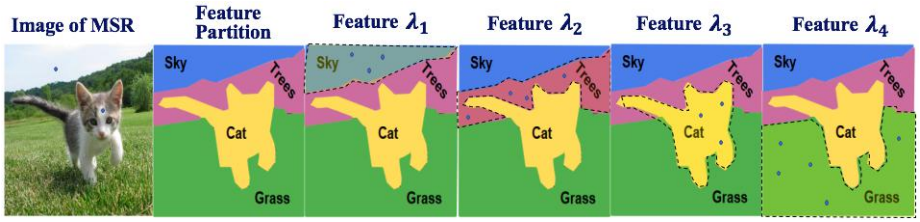


Figure 5: Illustration of the *maximum safe radius* (MSR) and *feature robustness* ($\mathtt{FR}_\Lambda$) problems. From left to right: an adversarial example with two pixel changes, feature extraction of the image, adversarial examples with three changed pixels on features 'Sky' and 'Cat', four changed pixels on 'Trees', and five pixel manipulations on 'Grass', respectively.

seen as a special case of the $\mathtt{FR}_\Lambda(k, d, \alpha, c)$ problem, when we let $|\Lambda(\alpha)| = 1$. We study them separately, because the $\mathtt{MSR}(k, d, \alpha, c)$ problem is interesting on its own, and, more importantly, we show later that they can be solved using different methods.

One can also consider a simpler variant of this problem, which aims to find a subset of features that are most resilient to perturbations, and which can be solved by only considering singleton sets of features. We omit the formalisation for reasons of space.

We illustrate the two problems, the *maximum safe radius* (MSR) and (the simpler variant of) *feature robustness* ($\mathtt{FR}'_\Lambda$), through Example 1.

Example 1. *As shown in Figure 5, the minimum distance from the original image to an adversary is two pixels, i.e.,* $\mathtt{MSR} = 2$ *(for simplicity here we take the* L_0*-norm). That is, for a norm ball with radius less than 2, the image is absolutely safe. Note that, for* MSR*, the manipulations can span different features. After feature extraction, we find the* maximum safe radius *of each feature, i.e.,* $\mathtt{MSR}_{\lambda_1} = 3$, $\mathtt{MSR}_{\lambda_2} = 4$, $\mathtt{MSR}_{\lambda_3} = 3$, $\mathtt{MSR}_{\lambda_4} = 5$.

Assume that we have a norm ball of radius d*, and a distance budget* d'*, then:*

- *if* $d = 4$*, then by definition we have* $\mathtt{FR}'_\Lambda = 4^\epsilon$*, i.e., manipulating 'Grass' cannot change the classification;*
- *if* $d = 10$ *and* $d' = 7$ *then we have* $\mathtt{FR}'_\Lambda = 5 < d' < d$*, i.e., all the features are fragile;*
- *if* $d = 10$ *and* $d' = 4$ *then* $d' < \mathtt{FR}'_\Lambda = 5 < d$*, i.e., the existence of an adversary is controllable by restricting perturbations to 'Grass'.*

Approximation Based on Finite Optimisation. Similarly to the case of the maximum safe radius, we reduce the feature robustness problem to finite optimisation by implementing the search for adversarial examples using input manipulations.

Definition 12. *Let* $\tau \in (0, 1]$ *be a manipulation magnitude. The finite feature robustness problem* $\mathtt{FFR}_\Lambda(\tau, k, d, \alpha, c)$ *based on input manipulation is as follows:*

$$\mathtt{FFR}_\Lambda(\tau, k, d, \alpha, c) = \max_{\lambda \in \Lambda(\alpha)} \{\mathtt{xFFR}_\Lambda(\lambda, \tau, k, d, \alpha, c)\} \tag{15}$$

where $\texttt{xFFR}_\Lambda(\lambda_m, \tau, k, d, \alpha_m, c) =$

$$\begin{cases} \min\limits_{X \subseteq P_{\lambda_m}} \min\limits_{\psi \in \Psi} \{||\alpha_m - \delta_{\tau,X,\psi}(\alpha_m)||_k + \texttt{FFR}_\Lambda(k, d, \delta_{\tau,X,\psi}(\alpha_m), c)\}, & \text{if } \alpha_m \notin adv_{k,d}(\alpha, c) \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

where Λ *is a feature extraction function, and* $\alpha_m, \delta_{\tau,X,\psi}(\alpha_m), m \in \mathbb{N}$, *are the perturbed inputs before and after the application of manipulation* $\delta_{\tau,X,\psi}$ *on a feature* λ_m*, respectively. If after selecting a feature* λ_m *no adversarial example can be reached, i.e.,* $\forall X \subseteq P_{\lambda_m} \forall \psi \in \Psi : \delta_{\tau,X,\psi}(\alpha_m) \notin adv_{k,d}(\alpha, c)$*, then we let* $\texttt{xFFR}_\Lambda(\lambda_m, \tau, k, d, \alpha_m, c) = d^\epsilon$.

Compared to Definition 11, the search for another input by $\min_{\alpha_{m+1} \in \eta(\alpha, L_k, d)}$ is implemented by combinatorial search over the finite sets of feature sets and instructions.

Error Bounds. The case for the feature robustness problem largely follows that of the maximum safe radius problem. First of all, we have the following lemma which bounds the error of $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c)$ to $\frac{1}{2}d(k, \tau)$, which depends on the value of magnitude.

Lemma 7. *If all* τ*-grid inputs are misclassification aggregators with respect to* $\frac{1}{2}d(k, \tau)$*, then* $\texttt{FR}_\Lambda(k, d, \alpha, c) \geq \texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) - \frac{1}{2}d(k, \tau)$.

Proof: We prove by contradiction. Assume that $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) = d'$ for some $d' > 0$, and $\texttt{FR}_\Lambda(k, d, \alpha, c) < d' - \frac{1}{2}d(k, \tau)$. Then, for all subsets $\Lambda \subseteq \Lambda(\alpha)$ of features, either for all $X \subseteq \bigcup_{\lambda \in \Lambda} P_\lambda$ and $\psi \in \Psi$ we have $\delta_{\tau,X,\psi}(\alpha) \notin adv_{k,d}(\alpha, c)$, or there must exist $X \subseteq \bigcup_{\lambda \in \Lambda} P_\lambda$ and $\psi \in \Psi$ such that

$$\alpha' = \delta_{\tau,X,\psi}(\alpha) \in adv_{k,d}(\alpha, c) \text{ and } ||\alpha' - \alpha||_k < d' - \frac{1}{2}d(k, \tau), \tag{17}$$

and α' is not a τ-grid input.

For the latter case, by Lemma 4, there must exist a τ-grid input α'' such that $\alpha' \in \eta(\alpha'', L_k, \frac{1}{2}d(k, \tau))$. Now because all τ-grid inputs are misclassification aggregators with respect to $\frac{1}{2}d(k, \tau)$, we have $\alpha'' \in adv_{k,d}(\alpha, c)$. By $\alpha'' \in adv_{k,d}(\alpha, c)$ and the fact that α'' is a τ-grid input, we have that

$$||\alpha - \alpha''||_k \leq ||\alpha - \alpha'||_k + \frac{1}{2}d(k, \tau). \tag{18}$$

Therefore, we have $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) < d'$ by the combining Equations (17) and (18). However, this contradicts the hypothesis that $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) = d'$.

For the former case, we have $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) = d' > d$. If $\texttt{FR}_\Lambda(k, d, \alpha, c) < d' - \frac{1}{2}d(k, \tau)$, then there exists an α' such that $\alpha' \in \eta(\alpha'', L_k, \frac{1}{2}d(k, \tau))$ for some τ-grid input α''. By the definition of misclassification aggregator, we have $\alpha'' \in adv_{k,d}(\alpha, c)$. This contradicts the hypothesis that $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) = d' > d$. □

Combining Lemmas 3, 6, and 7, we have the following theorem which shows that the reduction has a provable guarantee under the assumption of Lipschitz continuity. The approximation error depends linearly on the prediction confidence on discretised 'grid' inputs and is inversely proportional with respect to the Lipschitz constants of the network.

Theorem 2. *Let N be a Lipschitz network with a Lipschitz constant $\hbar_c$ for every class $c \in C$. If $d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})}$ for all τ-grid inputs $\alpha' \in G(\alpha, k, d)$, then we can use $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c)$ to estimate $\texttt{FR}_\Lambda(k, d, \alpha, c)$ with an error bound $\frac{1}{2}d(k,\tau)$.*

Proof: By Lemma 3, we have $\texttt{FR}_\Lambda(k, d, \alpha, c) \leq \texttt{FFR}_\Lambda(\tau, k, d, \alpha, c)$ for any $\tau > 0$. By Lemma 6 and Lemma 7, when $\texttt{FFR}_\Lambda(\tau, k, d, \alpha, c) = d'$, we have $\texttt{FR}_\Lambda(k, d, \alpha, c) \geq d' - \frac{1}{2}d(k,\tau)$, under the condition that $d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})}$ for all τ-grid inputs $\alpha' \in G(\alpha, k, d)$. Therefore, when $d(k,\tau) \leq \frac{2g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})}$ for all τ-grid inputs $\alpha' \in G(\alpha, k, d)$, if we use d' to estimate $\texttt{FR}_\Lambda(k, d, \alpha, c)$, we will have $d' - \frac{1}{2}d(k,\tau) \leq \texttt{FR}_\Lambda(k, d, \alpha, c) \leq d'$, i.e., the error bound is $\frac{1}{2}d(k,\tau)$. □

4. A Game-Based Approximate Verification Approach

In this section, we define a two-player game and show that the solutions of the two finite optimisation problems, $\texttt{FMSR}(k, d, \alpha, c)$ and $\texttt{FFR}_\Lambda(k, d, \alpha, c)$, given in Expressions (4) and (15) can be reduced to the computation of the rewards of Player I taking an optimal strategy. The two problems differ in that they induce games in which the two players are *cooperative* or *competitive*, respectively.

The game proceeds by constructing a sequence of atomic input manipulations to implement the optimisation objectives in Equations (4) and (15).

4.1. Problem Solving as a Two-Player Turn-Based Game

The game has two players, who take turns to act. Player I selects features and Player II then selects an atomic input manipulation within the selected features. While Player II aims to minimise the distance to an adversarial example, depending on the optimisation objective designed for either $\texttt{FMSR}(k, d, \alpha, c)$ or $\texttt{FFR}_\Lambda(k, d, \alpha, c)$, Player I can be cooperative or competitive. We remark that, in contrast to [23] where the games were originally introduced, we do not consider the nature player.

Formally, we let $M(k, d, \alpha, c) = (S \cup (S \times \Lambda(\alpha)), s_0, \{T_a\}_{a \in \{\texttt{I}, \texttt{II}\}}, L)$ be a game model, where

- S is a set of game states belonging to Player I such that each state represents an input in $\eta(\alpha, L_k, d)$, and $S \times \Lambda(\alpha)$ is a set of game states belonging to Player II where $\Lambda(\alpha)$ is a set of features of input α. We write $\alpha(s)$ for the input associated to the state $s \in S$.

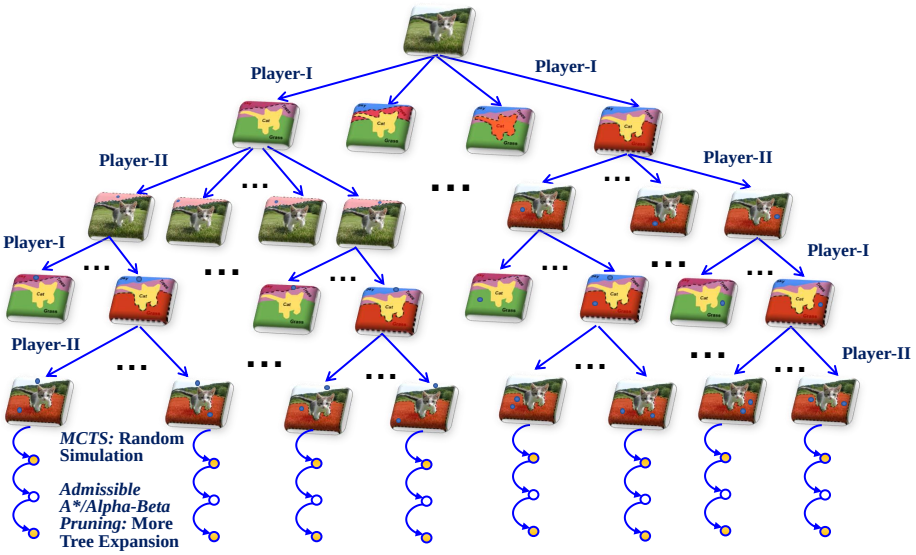


Figure 6: Two-player turn-based solution for finite optimisation. Player I selects features and Player II then performs an atomic input manipulation within the selected features. For the *Maximum Safe Radius* problem, both Player I and Player II aim to minimise the distance to an adversarial example; for the *Feature Robustness* problem, while Player II has the same objective, Player I plays against this, i.e., aiming to prevent the reaching of adversarial examples by taking suitable actions. The game terminates when an adversary is found or the distance budget for adversarial perturbation has been reached.

- $s_0 \in S$ is the initial game state such that $\alpha(s_0)$ is the original input α.
- The transition relation $T_{\mathtt{I}} : S \times \Lambda(\alpha) \to S \times \Lambda(\alpha)$ is defined as

$$T_{\mathtt{I}}(s, \lambda) = (s, \lambda), \tag{19}$$

  and transition relation $T_{\mathtt{II}} : (S \times \Lambda(\alpha)) \times \mathcal{P}(P_0) \times \Psi \to S$ is defined as

$$T_{\mathtt{II}}((s, \lambda), X, \psi) = \delta_{\tau, X, \psi}(\alpha(s)), \tag{20}$$

  where $X \subseteq P_\lambda$ is a set of input dimensions within feature λ, $\psi : P_0 \to \{-1, +1\}$ is a manipulation instruction, and $\delta_{\tau,X,\psi}$ is an atomic dimension manipulation as defined in Definition 3. Intuitively, in every game state $s \in S$, Player I will choose a feature λ, and, in response to this, Player II will choose an atomic input manipulation $\delta_{\tau,X,\psi}$.
- The labelling function $L : S \cup (S \times \Lambda(\alpha)) \to C$ assigns to each state s or (s, λ) a class $N(\alpha(s))$.

Figure 6 illustrates the game model with a partially-expanded game tree.

Strategy Profile. A path (or game play) of the game model is a sequence $s_1 u_1 s_2 u_2 ...$ of game states such that, for all $k \geq 1$, we have $u_k = T_{\mathtt{I}}(s_k, \lambda_k)$ for some feature

λ_k and $s_{k+1} = T_{\mathtt{II}}((s_k, \lambda_k), X_k, \psi_k)$ for some (X_k, ψ_k). Let $last(\rho)$ be the last state of a finite path ρ, and $Path_a^F$ be the set of finite paths such that $last(\rho)$ belongs to player $a \in \{\mathtt{I}, \mathtt{II}\}$.

Definition 13. *A stochastic strategy $\sigma_{\mathtt{I}} : Path_{\mathtt{I}}^F \to \mathcal{D}(\Lambda(\alpha))$ of Player $\mathtt{I}$ maps each finite path to a distribution over the next actions, and similarly for $\sigma_{\mathtt{II}} : Path_{\mathtt{II}}^F \to \mathcal{D}(\mathcal{P}(P_0) \times \mathcal{I})$ for Player $\mathtt{II}$. We call $\sigma = (\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}})$ a strategy profile.*

A strategy σ is deterministic if $\sigma(\rho)$ is a Dirac distribution, and is memoryless if $\sigma(\rho) = \sigma(last(\rho))$ for all finite paths ρ.

Rewards. We define a reward $R(\sigma, \rho)$ for a given strategy profile $\sigma = (\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}})$ and a finite path $\rho \in \bigcup_{a \in \{\mathtt{I}, \mathtt{II}\}} Path_a^F$. The idea of the reward is to accumulate the distance to the adversarial example found over a path. Note that, given σ, the game becomes a deterministic system. Let $\alpha'_\rho = \alpha(last(\rho))$ be the input associated with the last state of the path ρ. We write

$$t(\rho) \equiv (N(\alpha'_\rho) = c) \vee (||\alpha'_\rho - \alpha||_k > d), \tag{21}$$

representing that the path has reached a state whose associated input either is in the target class c or lies outside the region $\eta(\alpha, L_k, d)$. The path ρ can be terminated whenever $t(\rho)$ is satisfied. It is not hard to see that, due to the constraints in Definition 5, every infinite path has a finite prefix which can be terminated (that is, either when an adversarial example is found or the distance to the original image has exceeded d). During each expansion of the game model, an atomic manipulation is employed, which excludes the possibility that an input dimension is perturbed in smaller and smaller steps.

Definition 14. *Given a strategy profile $\sigma = (\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}})$ and a finite path ρ, we define a reward function as follows: $R(\sigma, \rho) =$*

$$\begin{cases} ||\alpha'_\rho - \alpha||_k, & \text{if } t(\rho) \text{ and } \rho \in Path_{\mathtt{I}}^F \\ \displaystyle\sum_{\lambda \in \Lambda(\alpha)} \sigma_{\mathtt{I}}(\rho)(\lambda) \cdot R(\sigma, \rho T_{\mathtt{I}}(last(\rho), \lambda)), & \text{if } \neg t(\rho) \text{ and } \rho \in Path_{\mathtt{I}}^F \\ \displaystyle\sum_{(X,\psi) \in \mathcal{P}(P_0) \times \Psi} \sigma_{\mathtt{II}}(\rho)(X, \psi) \cdot R(\sigma, \rho T_{\mathtt{II}}(last(\rho), X, \psi)), & \text{if } \rho \in Path_{\mathtt{II}}^F \end{cases} \tag{22}$$

where $\sigma_{\mathtt{I}}(\rho)(\lambda)$ is the probability of selecting feature λ on finite path ρ by Player $\mathtt{I}$, and $\sigma_{\mathtt{II}}(\rho)(X, \psi)$ is the probability of selecting atomic input manipulation $\delta_{\tau,X,\psi}$ based on ρ by Player $\mathtt{II}$. The expression $\rho T_{\mathtt{I}}(last(\rho), \lambda)$ is the resulting path of Player $\mathtt{I}$ selecting λ, and $\rho T_{\mathtt{II}}(last(\rho), X, \psi)$ is the resulting path of Player $\mathtt{II}$ applying $\delta_{\tau,X,\psi}$ on α'_ρ. We note that a path only terminates on Player $\mathtt{I}$ states.

Intuitively, if an adversarial example is found then the reward assigned is the distance to the original input, otherwise it is the weighted summation of the rewards of its children.

Players' Objectives. Players' strategies are to maximise their rewards in a game. The following players' objectives are designed to match the finite optimisation problems stated in Equations (4) and (15).

Definition 15. *In a game, Player* II *chooses a strategy* $\sigma_{\mathtt{II}}$ *to minimise the reward* $R((\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}}), s_0)$*, whilst Player* I *has different goals depending on the optimisation problem under consideration.*

- *For the* maximum safe radius *problem, Player* I *chooses a strategy* $\sigma_{\mathtt{I}}$ *to minimise the reward* $R((\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}}), s_0)$*, based on the strategy* $\sigma_{\mathtt{II}}$ *of Player* II*. That is, the two players are* cooperative*.*
- *For the* feature robustness *problem, Player* I *chooses a strategy* $\sigma_{\mathtt{I}}$ *to maximise* $R((\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}}), s_0)$*, based on the strategy* $\sigma_{\mathtt{II}}$ *of Player* II*. That is, the two players are* competitive*.*

The goal of the game is for Player I to choose a strategy $\sigma_{\mathtt{I}}$ to optimise its objective, to be formalised below.

4.2. Safety Guarantees via Optimal Strategy

For different objectives $x \in \{\mathtt{MSR}(k, d, \alpha, c), \mathtt{FR}_{\Lambda}(k, d, \alpha, c)\}$ of Player I, we construct different games. Given a game model $M(k, d, \alpha, c)$ and an objective x of Player I, there exists an *optimal strategy profile* $\sigma = (\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}})$, obtained by both players optimising their objectives. We will consider the algorithms to compute the optimal strategy profile in Section 5. Here we focus on whether the obtained optimal strategy profile σ is able to implement the finite optimisation problems in Equations (4) and (15).

First of all, we formally define the goal of the game.

Definition 16. *Given a game model* $M(k, d, \alpha, c)$*, an objective* x *of Player* I*, and an optimal strategy profile* $\sigma = (\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}})$*, the goal of the game is to compute the value*

$$val(M(k, d, \alpha, c), x) = R(\sigma, s_0) \tag{23}$$

That is, the goal is to compute the reward of the initial state s_0 *based on* σ*. Note that an initial state* s_0 *is also a finite path, and it is a Player* I *state.*

We have the following Theorems 3 and 4 to confirm that the game can return the optimal values for the two finite optimisation problems.

Theorem 3. *Assume that Player* I *has the objective* $\mathtt{MSR}(k, d, \alpha, c)$*. Then*

$$val(M(k, d, \alpha, c), \mathtt{MSR}(k, d, \alpha, c)) = \mathtt{FMSR}(\tau, k, d, \alpha, c) \tag{24}$$

Proof: First, we show that $||\alpha - \alpha'||_k \geq val(M(k, d, \alpha, c), \mathtt{MSR}(k, d, \alpha, c))$ for any input α' such that $\alpha' \in \eta(\alpha, L_k, d)$, $\alpha' \in adv_{k,d}(\alpha, c)$, and α' is a τ-grid input. Intuitively, it says that Player I reward from the game on the initial state s_0 is no greater than the distance to any τ-grid adversarial example. That

is, once computed, the $val(M(k,d,\alpha,c),\texttt{MSR}(k,d,\alpha,c))$ is a lower bound of the optimisation problem $\texttt{FMSR}(\tau,k,d,\alpha,c)$. This can be obtained by the fact that every τ-grid input can be reached by some game play.

Second, from the termination condition of the game plays, we can see that if $val(M(k,d,\alpha,c),\texttt{MSR}(k,d,\alpha,c)) \leq ||\alpha-\alpha'||_k$ for some α' then there must exist some α'' such that $val(M(k,d,\alpha,c),\texttt{MSR}(k,d,\alpha,c)) = ||\alpha-\alpha''||_k$. Therefore, we have that $val(M(k,d,\alpha,c),\texttt{MSR}(k,d,\alpha,c))$ is the minimum value of $||\alpha-\alpha'||_k$ among all α' with $\alpha' \in \eta(\alpha,L_k,d)$, $\alpha' \in adv_{k,d}(\alpha,c)$, and α' is a τ-grid input.

Finally, we notice that the above minimum value of $||\alpha-\alpha'||_k$ is equivalent to the optimal value required by Equation (4). □

Theorem 4. *Assume that Player* I *has the objective* $\texttt{FR}_\Lambda(k,d,\alpha,c)$. *Then*

$$val(M(k,d,\alpha,c),\texttt{FR}_\Lambda(k,d,\alpha,c)) = \texttt{FFR}_\Lambda(\tau,k,d,\alpha,c) \tag{25}$$

Proof: First of all, let Λ_1 be the set of features and Δ_1 be the set of atomic input manipulations in achieving the optimal value of $\texttt{FFR}_\Lambda(\tau,k,d,\alpha,c)$. We can construct a game play for (Λ_1,Δ_1). More specifically, the game play leads from the initial state to a terminal state, by recursively selecting an unused input manipulation and its associated feature and defining the corresponding moves for Player I and Player II, respectively. Therefore, because the strategy profile σ is optimal, we have $val(M(k,d,\alpha,c),\texttt{FR}_\Lambda(k,d,\alpha,c)) \geq \texttt{FFR}_\Lambda(\tau,k,d,\alpha,c)$.

On the other hand, we notice that the ordering of the applications of atomic input manipulations does not matter, because the reward of the terminal state is the distance from its associated input to the original input. Therefore, because the game explores exactly all the possible applications of atomic input manipulations and $\texttt{FFR}_\Lambda(\tau,k,d,\alpha,c)$ is the optimal value by its definition, by Lemma 2 we have that $val(M(k,d,\alpha,c),\texttt{FR}_\Lambda(k,d,\alpha,c)) \leq \texttt{FFR}_\Lambda(\tau,k,d,\alpha,c)$. □

Combining Theorems 3, 4 with Theorems 1, 2, we have the following corollary, which states that the optimal game strategy is able to achieve the optimal value for the maximum safe radius problem $\texttt{MSR}(k,d,\alpha,c)$ and the feature robustness problem $\texttt{FR}_\Lambda(k,d,\alpha,c)$ with an error bound $\frac{1}{2}d(k,\tau)$.

Corollary 1. *The two-player turn-based game is able to solve the* maximum safe radius *problem of Equation (3) and the* feature robustness *problem of Equation (12) with an error bound* $\frac{1}{2}d(k,\tau)$, *when the magnitude* τ *is such that* $d(k,\tau) \leq \frac{2g(\alpha',N(\alpha'))}{\max_{c'\in C,c'\neq N(\alpha')}(\hbar_{N(\alpha')}+\hbar_{c'})}$ *for all* τ*-grid inputs* $\alpha' \in G(\alpha,k,d)$.

Furthermore, we have the following lemma.

Lemma 8. *For game* $M(k,d,\alpha,c)$ *with goal* $val(M(k,d,\alpha,c),\texttt{MSR}(k,d,\alpha,c))$, *deterministic and memoryless strategies suffice for Player* I, *and similarly for* $M(k,d,\alpha,c)$ *with goal* $val(M(k,d,\alpha,c),\texttt{FR}_\Lambda(k,d,\alpha,c))$.

4.3. Complexity of the Problem

As a by-product of Lemma 8, the theoretical complexity of the problems is in PTIME, with respect to the size of the game model $M(k,d,\alpha,c)$. However, the

size of the game is exponential with respect to the number of input dimensions. More specifically, we have the following complexity result with respect to the manipulation magnitude τ, the pre-specified range size d, and the number of input dimensions n.

Theorem 5. *Given a game $M(k, d, \alpha, c)$, the computational time needed for the value $val(M(k, d, \alpha, c), x)$, where $x \in \{\mathtt{MSR}(k, d, \alpha, c), \mathtt{FR}_\Lambda(k, d, \alpha, c)\}$, is polynomial with respect to d/τ and exponential with respect to n.*

Proof: We can see that the size of the grid, measured as the number $|G(\alpha, k, d)|$ of τ-grid inputs in $\eta(\alpha, L_k, d)$, is polynomial with respect to d/τ and exponential with respect to n. From a τ-grid to any of its neighbouring τ-grids, each player needs to take a move. Therefore, the number of game states is doubled (i.e., polynomial) over $|G(\alpha, k, d)|$. This yields PTIME complexity of solving the game. □

Considering that the problem instances we work with usually have a large input dimensionality, this complexity suggests that directly working with the explicit game models is impractical. If we consider an alternative representation of a game tree (i.e., an unfolded game model) of finite depth to express the complexity, the number of nodes on the tree is $O(n^h)$ for h the length of the longest finite path without a terminating state. While the precise size of $O(n^h)$ is dependent on the problem (including the image α and the difficulty of crafting an adversarial example), it is roughly $O(50000^{100})$ for the images used in the ImageNet competition and $O(1000^{20})$ for smaller images such as GTSRB, CIFAR10, and MNIST. This is beyond the capability of existing approaches for exact or ϵ-approximate computation of probability (e.g., reduction to linear programming [29], value iteration, and policy iteration, etc.) that are used in probabilistic verification.

5. Algorithms and Implementation

In this section we describe the implementation of the game-based approach introduced in this paper. Figure 7 presents an overview of the reductions from the original problems to the solution of a two-player game for the case of Lipschitz networks, described in Section 3. Because exact computation of optimal rewards is computationally hard, we approximate the rewards by means of algorithms that unfold the game tree based on Monte Carlo tree search (MCTS), Admissible A*, and Alpha-Beta Pruning.

We take a principled approach to compute for each of the two game values, $\mathtt{MSR}(k, d, \alpha, c)$ and $\mathtt{FR}_\Lambda(k, d, \alpha, c)$, an upper bound and a lower bound. Our algorithms can gradually, but strictly, improve the bounds, so that they gradually converge as the computation proceeds. For $x \in \{\mathtt{MSR}(k, d, \alpha, c), \mathtt{FR}_\Lambda(k, d, \alpha, c)\}$, we write l_x and u_x for their lower and upper bound, respectively. The bounds can be interesting in their own. For example, a lower bound $\mathsf{l}_{\mathtt{MSR}(k,d,\alpha,c)}$ suggests absolute safety of an L_k norm ball with radius $\mathsf{l}_{\mathtt{MSR}(k,d,\alpha,c)}$ from the original input α, and an upper bound $\mathsf{u}_{\mathtt{MSR}(k,d,\alpha,c)}$ suggests the existence of an adversarial

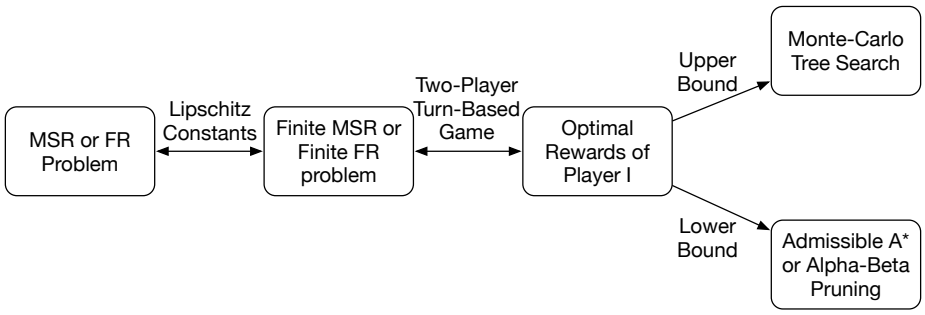


Figure 7: A game-based approximate verification approach for the *Maximum Safe Radius* and *Feature Robustness* problems.

example α' such that $||\alpha - \alpha'||_k = \mathsf{u}_{\mathtt{MSR}(k,d,\alpha,c)}$. On the other hand, given distance budget d', $\mathsf{u}_{\mathtt{FR}_\Lambda(k,d,\alpha,c)} \leq d'$ indicates an unsafe distance from which the existence of adversarial examples is not controllable.

We consider two feature extraction approaches to generate feature partitioning: a saliency-guided grey-box approach and a feature-guided black-box approach. For the grey-box approach, adapted from [17], each dimension of an input is evaluated on its sensitivity to the classification outcome of a DNN. For the black-box procedure, Scale Invariant Feature Transform (SIFT) [18] is used to extract image features, based on which a partition is computed. SIFT is a reasonable proxy for human perception, irrelevant to any image classifier, and its extracted features have a set of invariant characteristics, such as scaling, rotation, translation, and local geometric distortion. Readers are referred to Section 6.1 for an experimental illustrations of these two approaches.

Next we present the algorithms we employ to compute the upper and lower bounds of the values of the games, as well as their convergence analysis.

5.1. Upper Bounds: Monte Carlo Tree Search (MCTS)

We present an approach based on Monte Carlo tree search (MCTS) [30] to find an optimal strategy asymptotically. As a heuristic search algorithm for decision processes notably employed in game play, MCTS focuses on analysing the most promising moves via expanding the search tree based on random sampling of the search space. The algorithm, whose pseudo-code is presented in Algorithm 1, gradually expands a *partial game tree* by sampling the strategy space of the model $M(k, d, \alpha, c)$. With the upper confidence bound (UCB) [31] as the exploration-exploitation trade-off, MCTS has a theoretical guarantee that it converges to the optimal solution when the game tree is fully explored. In the following, we explain the components of the algorithm.

Concerning the data structure, we maintain the set of nodes on the partial tree $\mathcal{T}(k, d, \alpha, c)$. For every node o on the partial tree, we maintain three variables, r_o, n_o, e_o, which represent the accumulated reward, the number of visits, and the current best input with respect to the objective of the player, respectively. We remark that e_o is usually different from $\alpha(s_o)$, which is the input

Algorithm 1 Monte Carlo Tree Search for DNN Verification

1: **Input:** A game model $M(k, d, \alpha, c)$, a termination condition tc
2: **Output:** $val(M(k, d, \alpha, c), \mathtt{FR}_\Lambda(k, d, \alpha, c))$ or $val(M(k, d, \alpha, c), \mathtt{MSR}(k, d, \alpha, c))$
3: **procedure** MCTS($M(k, d, \alpha, c), tc$)
4: $root \leftarrow s_0$
5: **While**($\neg tc$):
6: $leaf \leftarrow selection(root)$
7: $newnodes \leftarrow expansion(M(k, d, \alpha, c), leaf)$
8: **for** *node* in *newnodes*:
9: $v \leftarrow Simulation(M(k, d, \alpha, c), node)$
10: $backPropogation(node, v)$
11: **return** optimal value of the *root* node

associated with the game state s_o. Moreover, for every node o, we record its parent node p_o and a set $\mathcal{C}_o$ of its children nodes. The value $val(M(k, d, \alpha, c), x)$ of the game is approximated by $||e_{root} - \alpha||_k$, which represents the distance between the original input and the current best input maintained by the root node of the tree.

The *selection* procedure starts from the *root* node, which contains the original image, and conducts a tree traversal until reaching a $leaf$ node (Line 6). From a node, the next child node to be selected is dependent on an exploration-exploitation balance, i.e., UCB [31]. More specifically, on a node o, for every child node $o' \in \mathcal{C}_o$, we let

$$v(o, o') = \frac{d * n_{o'}}{r_{o'}} + \sqrt{\frac{2 \ln n_o}{n_{o'}}} \tag{26}$$

be the weight of choosing o' as the next node from o. Then the actual choice of a next node is conducted by sampling over a probabilistic distribution $Prob_o : \mathcal{C}_o \to [0, 1]$ such that

$$Prob_o(o') = \frac{v_{o,o'}}{\sum_{o' \in \mathcal{C}_o} v_{o,o'}} \tag{27}$$

which is a normalisation over the weights of all children. On a $leaf$ node o, the *expansion* procedure returns a set of children nodes $\mathcal{C}_o$ by applying the transition relation in the game model $M(k, d, \alpha, c)$ (Line 7). These new nodes are added into the partial tree $\mathcal{T}(k, d, \alpha, c)$. This is the only way for the partial tree to grow. After expanding the leaf node to have its children added to the partial tree, we call the *Simulation* procedure on every child node (Line 9). A simulation on a new node o is a play of the game from o until it terminates. Players act randomly during the simulation. Every simulation terminates when reaching a terminal node α'. Once a terminal node α' is reached, a reward $||\alpha - \alpha'||_k$ can be computed. This reward, together with the input α', is then *backpropagated* from the new child node through its ancestors until reaching the root (Line 10). Every time a new reward v is backpropagated through a node o, we update its associated reward r_o into $r_o + v$ and increase its number of visits into $n_o + 1$. The update of current best input e_o depends on the player

who owns the node. For the $\texttt{MSR}(k, d, \alpha, c)$ game, e_o is made equivalent to $e_{o'}$ such that

$$o' = \arg\min_{o_1 \in \mathcal{C}_o} ||\alpha - e_{o_1}||_k \tag{28}$$

For the $\texttt{FR}_\Lambda(k, d, \alpha, c)$ game, Player II also takes the above approach, i.e., Equation (28), to update e_o, but for Player I we let e_o be $e_{o''}$ such that

$$o'' = \arg\max_{o_1 \in \mathcal{C}_o} ||\alpha - e_{o_1}||_k \tag{29}$$

We remark the game is not zero-sum for the maximum safe radius problem.

5.2. Lower Bounds: Admissible A in a Cooperative Game*

To enable the computation of lower bounds with a guarantee, we consider algorithms which can compute optimal strategy deterministically, without relying on the asymptotic convergence as MCTS does. In this section, we exploit Admissible A* to achieve the lower bound of Player I reward when it is *cooperative*, i.e., $\texttt{MSR}(k, d, \alpha, c)$, and in Section 5.3 we use Alpha-Beta Pruning to obtain the lower bound of Player I reward when it is *competitive*, i.e., $\texttt{FR}_\Lambda(k, d, \alpha, c)$.

The A* algorithm gradually unfolds the game model into a tree. It maintains a set of leaf nodes of the unfolded partial tree, computes an estimate for every node in the set, and selects the node with the least estimated value to expand. The estimation consists of two components, one for the exact cost up to now and the other for the estimated cost of reaching the goal node. In our case, for each game state s, we assign an estimated distance value

$$\mathit{distances}(s) = ||\alpha(s) - \alpha(s_0)||_k + \mathit{heuristic}(\alpha(s)) \tag{30}$$

where the first component $||\alpha(s) - \alpha(s_0)||_k$ represents the distance from the initial state s_0 to the current state s, and the second component $\mathit{heuristic}(\alpha(s))$ denotes the estimated distance from the current state s to a terminal state.

An *admissible* heuristic function is to, given a current input, never overestimate the cost of reaching the terminal game state. Therefore, to achieve the lower bound, we need to take an admissible heuristic function. We remark that, if the heuristic function is inadmissible (i.e., does not guarantee the underestimation of the cost), then the A* algorithm cannot be used to compute the lower bound, but instead can be used to compute the upper bound.

We utilise the minimum confidence margin $g(\alpha', N(\alpha'))$ defined in Definition 10 to obtain an admissible heuristic function.

Lemma 9. *For any game state s such that $\alpha(s) = \alpha'$, the following heuristic function is admissible:*

$$\mathit{heuristic}(\alpha') = \frac{g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')}(\hbar_{N(\alpha')} + \hbar_{c'})} \tag{31}$$

Algorithm 2 Admissible A* for DNN Verification

1: **Input:** A game model $M(k, d, \alpha, c)$, a termination condition tc
2: **Output:** $val(M(k, d, \alpha, c), \texttt{MSR}(k, d, \alpha, c))$
3: **procedure** AdmissibleA*$(M(k, d, \alpha, c), tc)$
4: $root \leftarrow s_0$
5: **While**$(\neg tc)$:
6: $features \leftarrow$ Player `I` ($root$, feature extraction = grey/black)
7: **for** $feature$ in $features$:
8: $dimensions \leftarrow$ Player `II` ($feature$)
9: $newnodes \leftarrow AtomicManipulation(dimensions)$
10: **for** $node$ in $newnodes$:
11: $distances \leftarrow DistanceEstimation(node)$
12: $root \leftarrow MaximumSafeRadius(distances)$
13: **return** $||\alpha(root) - \alpha(s_0)||_k$

Proof: Consider the expression $g(\alpha', N(\alpha')) - g(\alpha'', N(\alpha'))$, where α' is the current state and α'' is the last state before a terminal state. Then we have that

$$g(\alpha', N(\alpha')) - g(\alpha'', N(\alpha')) \leq g(\alpha', N(\alpha')) \tag{32}$$

Now because

$$\begin{aligned}
&g(\alpha', N(\alpha')) - g(\alpha'', N(\alpha')) \\
&= \min_{c \in C, c \neq N(\alpha')} \{N(\alpha', N(\alpha')) - N(\alpha', c)\} - \min_{c' \in C, c' \neq N(\alpha')} \{N(\alpha'', N(\alpha')) - N(\alpha'', c')\} \\
&\leq \max_{c' \in C, c' \neq N(\alpha')} \{|N(\alpha', N(\alpha')) - N(\alpha'', N(\alpha'))| + |N(\alpha'', c') - N(\alpha', c')|\} \\
&\leq \max_{c' \in C, c' \neq N(\alpha')} (\hbar_{N(\alpha')} + \hbar_{c'}) ||\alpha' - \alpha''||_k
\end{aligned} \tag{33}$$

we can let

$$\max_{c' \in C, c' \neq N(\alpha')} (\hbar_{N(\alpha')} + \hbar_j) ||\alpha' - \alpha''||_k \leq g(\alpha', N(\alpha')) \tag{34}$$

Thus, we define

$$heuristic(\alpha') = \frac{g(\alpha', N(\alpha'))}{\max_{c' \in C, c' \neq N(\alpha')} (\hbar_{N(\alpha')} + \hbar_{c'})} \tag{35}$$

which is sufficient to ensure that $g(\alpha'', N(\alpha')) \geq 0$ for any α''. That is, the distance $heuristic(\alpha')$ is a lower bound of reaching a misclassification. □

The Admissible A* algorithm is presented in Algorithm 2. In the following, we explain the main components of the algorithm. For each *root* node (initialised as the original input), Player `I` chooses between mutually exclusive $features$ partitioned based on either the grey-box or black-box approach. Subsequently, in each $feature$, Player `II` chooses among all the *dimensions* within each feature (Line 4-8). On each of the *dimensions*, an *AtomicManipulation* is constructed

and applied. We add $+\tau$ and $-\tau$ to each dimension, and make sure that it does not exceed the upper and lower bounds of the input dimension, e.g., 1 and 0 if the input is pre-processed (normalised). If exceeded, the bound value is used instead. This procedure essentially places adversarial perturbations on the image, and all manipulated images become the *newnodes* (Line 9). For each *node* in the *newnodes*, the $DistanceEstimation$ function in Equation (30) is used to compute a value, which is then added into the set *distances*. The set *distances* maintains the estimated values for all leaf nodes (Line 10-11). Among all the leaf nodes whose values are maintained in *distances*, we select the one with the minimum $MaximumSafeRadius$ as the new *root* (Line 12).

As for the termination condition $\neg tc$, the algorithm gradually unfolds the game tree with increasing tree depth $td = 1, 2, ...$ Because all nodes on the same level of the tree have the same distance to the original input α, every tree depth $td > 0$ is associated with a distance $d(td)$, such that $d(td)$ is the distance of the nodes at level td. For a given tree depth td, we have a termination condition $tc(td)$ requiring that either

- all the tree nodes up to depth td have been explored, or
- the current *root* is an adversarial example.

For the latter, $||\alpha(root) - \alpha(s_0)||_k$ is returned and the algorithm converges. For the former, we update $d(td)$ as the current lower bound of the game value $val(M(k, d, \alpha, c), \texttt{MSR}(k, d, \alpha, c))$. Note that the termination condition guarantees the closest adversarial example that corresponds to `FMSR`, which is within distance $\frac{1}{2}d(k, \tau)$ from the actual closest adversarial example corresponding to `MSR`.

5.3. Lower Bounds: Alpha-Beta Pruning in a Competitive Game

Alpha-Beta Pruning is an adversarial search algorithm, applied commonly in two-player games, to minimise the possible cost in a maximum cost scenario. In this paper, we apply Alpha-Beta Pruning to compute the lower bounds of Player `I` reward in a *competitive* game, i.e., $\texttt{FR}_\Lambda(k, d, \alpha, c)$.

Lemma 10. *For any game state $s \in S \cup (S \times \Lambda(\alpha))$, we let $\sigma_{\texttt{I}}(s) \in S \times \Lambda(\alpha)$ be the next state of $s \in S$ after Player `I` taking an action $\sigma_{\texttt{I}}$, and $\sigma_{\texttt{II}}(s) \in S$ be the next state of $s \in S \times \Lambda(\alpha)$ after Player `II` taking an action $\sigma_{\texttt{II}}$. If using $\texttt{alpha}(s)$ (initialised as $-\infty$) to denote Player `I` current maximum reward on state s and $\texttt{beta}(s)$ (initialised as $+\infty$) to denote Player `II` current minimum reward on state s, and let*

$$\texttt{alpha}(s) = \max_{\sigma_{\texttt{I}}} \texttt{beta}(\sigma_{\texttt{I}}(s)) \qquad \text{if } s \in S \tag{36}$$

$$\texttt{beta}(s) = \min_{\sigma_{\texttt{II}}} \texttt{alpha}(\sigma_{\texttt{II}}(s)) \qquad \text{if } s \in S \times \Lambda(\alpha) \tag{37}$$

then $\texttt{alpha}(s_0)$ is a lower bound of the value $val(M(k, d, \alpha, c), \texttt{FR}_\Lambda(k, d, \alpha, c))$.

Algorithm 3 Alpha-Beta Pruning for DNN Verification

1: **Input:** A game model $M(k, d, \alpha, c)$, a termination condition tc
2: **Output:** $val(M(k, d, \alpha, c), \texttt{FR}_\Lambda(k, d, \alpha, c))$
3: **procedure** AlphaBeta($M(k, d, \alpha, c), tc$)
4: $root \leftarrow s_0$
5: $root.\texttt{alpha} \leftarrow -\infty$
6: $features \leftarrow$ Player I ($root$, feature extraction = grey/black)
7: **for** $feature$ in $features$:
8: $feature.\texttt{beta} \leftarrow +\infty$
9: $dimensions \leftarrow$ Player II ($feature$)
10: $newnodes \leftarrow AtomicManipulation(dimensions)$
11: **for** $node$ in $newnodes$:
12: **if** tc: **return** $||\alpha(node) - \alpha(s_0)||_k$
13: **else**: $node.\texttt{alpha} \leftarrow$AlphaBeta($node, tc$)
14: $feature.\texttt{beta} \leftarrow \min(newnodes.\texttt{alpha})$
15: $root.\texttt{alpha} \leftarrow \max(features.\texttt{beta})$
16: **return** $root.\texttt{alpha}$

Note that, for a game state s, whenever $\texttt{alpha}(s) \geq \texttt{beta}(s')$ for some $s' = \sigma_{\texttt{I}}(s)$ is satisfied, Player I does not need to consider the remaining strategies of Player II on state s', as such will not affect the final result. This is the pruning of the game tree. The Alpha-Beta Pruning algorithm is presented in Algorithm 3. Many components of the algorithm are similar to those of Admissible A*, except that each node maintains two values: alpha value and beta value. For every node, its alpha value is initialised as $-\infty$ and its beta value is initialised as $+\infty$. For each $feature$, its beta value is the minimum of all the alpha values of the perturbed inputs whose manipulated dimensions are within this feature (Line 14); for $root$ in each recursion, the alpha value is the maximum of all the beta values of the features (Line 15). Intuitively, beta maintains the MSR of each feature, while alpha maintains the $\texttt{FR}_\Lambda$ of an input.

5.4. Anytime Convergence

In this section, we show the convergence of our approach, i.e., that both bounds are monotonically improved with respect to the optimal values.

Upper Bounds: $1/\epsilon$-Convergence and Practical Termination Condition (tc). Because we are working with a finite game, MCTS is guaranteed to converge when the game tree is fully expanded, but the worst case convergence time may be prohibitive. In practice, we can work with $1/\epsilon$-convergence by letting the program terminate when the current best bound has not been improved for e.g., $\lceil 1/\epsilon \rceil$ iterations, where $\epsilon > 0$ is a small real number. We can also impose time constraint tc, and ask the program to return once the elapsed time of the computation has exceeded tc.

In the following, we show that the intermediate results from Algorithm 1 can be the upper bounds of the optimal values, and the algorithm is continuously improving the upper bounds, until the optimal values are reached.

Lemma 11. *Let $||\alpha' - e_{root}||_k$ be the returned result from Algorithm 1. For an* $\texttt{FMSR}(k, d, \alpha, c)$ *game, we have that*

$$||\alpha' - e_{root}||_k \geq val(M(k, d, \alpha, c), \texttt{MSR}(k, d, \alpha, c)). \tag{38}$$

Moreover, the discrepancy between $||\alpha' - e_{root}||_k$ and $val(M(k, d, \alpha, c), \texttt{MSR}(k, d, \alpha, c))$ improves monotonically as the computation proceeds.

Proof: Assume that we have a partial tree $\mathcal{T}(k, d, \alpha, c)$. We prove by induction on the structure of the tree. As the base case, for each leaf node o we have that its best input e_o is such that

$$||\alpha - e_o||_k \geq val(M(k, d, \alpha, c), \texttt{MSR}(k, d, \alpha, c)) \tag{39}$$

because a random simulation can always return a current best, which is an upper bound to the global optimal value. The equivalence holds when the simulation found an adversarial example with minimum distance.

Now, for every internal node o, by Equation (28) we have that

$$\exists o_1 \in \mathcal{C}_o : ||\alpha - e_o||_k \geq ||\alpha - e_{o_1}||_k \tag{40}$$

which, together with Equation (39) and induction hypothesis, implies that $||\alpha - e_o||_k \geq val(M(k, d, \alpha, c), \texttt{MSR}(k, d, \alpha, c))$. Equation (38) holds since the root node is also an internal node.

The monotonic improvement can be seen from Equation (28), namely that, when, and only when, the discrepancy for the leaf node is improved after a new round of random simulation, can the discrepancy for the root node be improved. Otherwise, it remains the same. □

Similarly, we have the following lemma for the feature robustness game.

Lemma 12. *Let $||\alpha' - e_{root}||_k$ be the returned result from Algorithm 1. For an* $\texttt{FFR}_\Lambda(k, d, \alpha, c)$ *game, we have that*

$$||\alpha' - e_{root}||_k \geq val(M(k, d, \alpha, c), \texttt{FR}_\Lambda(k, d, \alpha, c)) \tag{41}$$

Proof: The proof is similar to that of Lemma 11, except that, according to Equation (29), for the nodes of Player $\texttt{I}$ (including the root node) to reduce the discrepancy, i.e., $||\alpha - e_o||_k - val(M(k, d, \alpha, c), \texttt{FR}_\Lambda(k, d, \alpha, c))$, it requires that all its children nodes reduce their discrepancy. □

Lower Bounds: Gradual Expansion of the Game Tree. The monotonicity of the lower bounds is achieved by gradually increasing the tree depth td. Because, in both algorithms, the termination conditions are the full exploration of the partial trees up to the depth td, it is straightforward that the results returned by the algorithms are either the lower bounds or the converged results.

6. Experimental Results

This section presents experimental results for the proposed game-based approach for safety verification of deep neural networks, focused on demonstrating convergence and comparison with state-of-the art techniques.

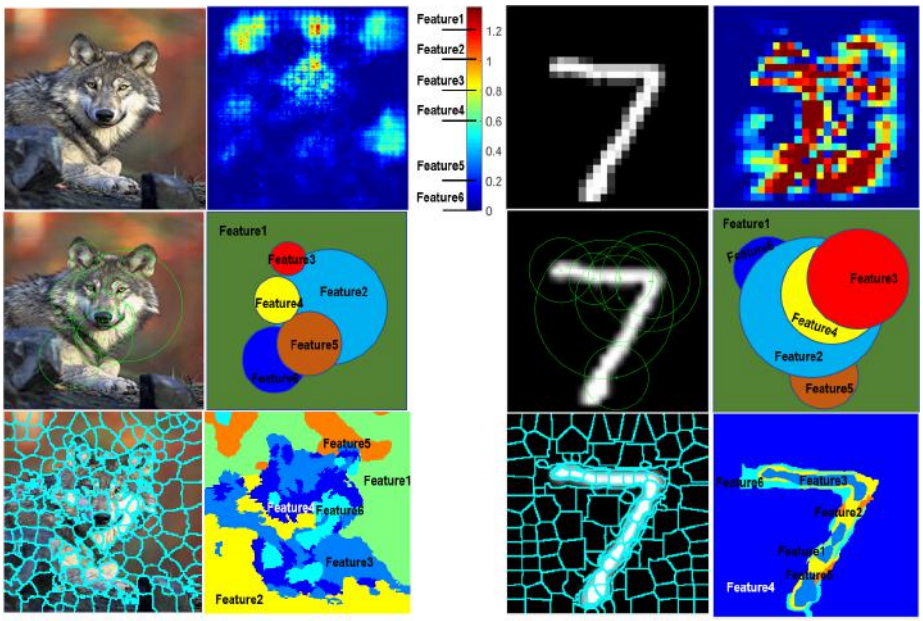


Figure 8: Examples of three feature extraction methods applied on the ImageNet (left) and MNIST (right) datasets, respectively. The top row: image segmentation via the saliency map generation method introduced in [17]. Middle row: feature extraction using the SIFT approach [18]. Bottom row: image partition using K-means clustering and superpixels [32].

6.1. Feature-Based Partitioning

Our game-based approach, where Player `I` determines features and Player `II` selects pixels or dimensions within the selected feature, requires an appropriate feature partitioning method into disjoint sets of dimensions. In Figure 8 we illustrate three distinct feature extraction procedures on a colour image from the ImageNet dataset and a grey-scale image from the MNIST dataset. Though we work with image classifier networks, our approach is flexible and can be adapted to a range of feature partitioning methods.

The first technique for image segmentation is based on the saliency map generated from an image classifier such as a DNN. As shown Figure 8 (top row), the heat-map is produced by quantifying how sensitive each pixel is to the classification outcome of the DNN. By ranking these sensitivities, we separate the pixels into a few disjoint sets. The second feature extraction approach, shown in Figure 8 (middle row), is independent of any image classifier, but instead focuses on abstracting the invariant properties directly from the image. Here we show segmentation results from the SIFT method [18], which is invariant to image translation, scaling, rotation, and local geometric distortion. More details on how to adapt SIFT for safety verification on DNNs can be found in [23]. The third feature extraction method is based on superpixel representation, a dimensionality reduction technique widely applied in various computer vision

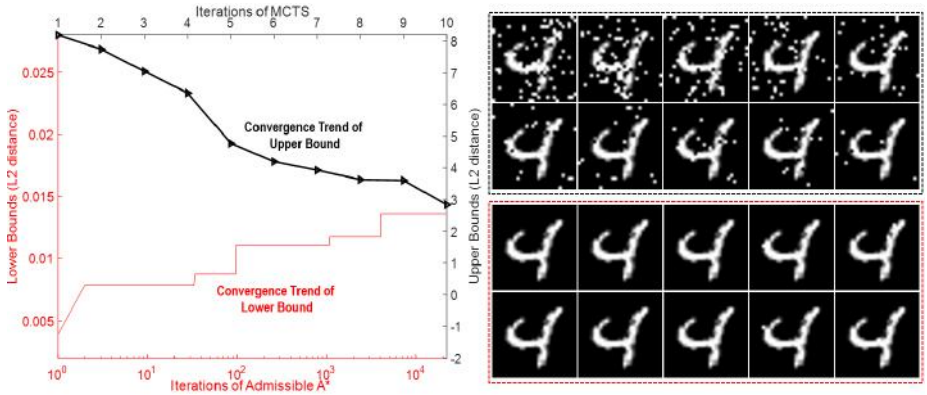


Figure 9: Convergence of *maximum safe radius* in a *cooperative* game with grey-box feature extraction of a MNIST image originally classified as "4". Left: The convergence trends of the upper bound from MCTS and the lower bound from Admissible A* for the `MSR` problem. Right: The generated *adversarial* images while searching for the upper bound via MCTS, and lower boundary *safe* images while searching for the lower bound via Admissible A*.

applications. Figure 8 (bottom row) demonstrates an example of how to generate superpixels (i.e., the pixel clusters marked by the green grids) using colour features and K-means clustering [32].

6.2. Lipschitz Constant Estimation

Our approach assumes knowledge of a (not necessarily tight) Lipschitz constant. Several techniques can be used to estimate such a constant, including FastLin/FastLip [33], Crown [34] and DeepGO [16]. For more information see the Related Work section.

The size of the Lipschitz constant is inversely proportional to the number of grid points and error bound, and therefore affects computational performance. We remark that, due to the high non-linearity and high-dimensionality of modern DNNs, it is non-trivial to conduct verification even if the Lipschitz constant is known.

6.3. Convergence Analysis of the Upper and Lower Bounds

We demonstrate convergence of the bound computation for the maximum safe radius and feature robustness problems, evaluated on standard benchmark datasets MNIST, CIFAR10, and GTSRB. The architectures of the corresponding trained neural networks as well as their accuracy rates can be found in Appendix A.1.

Convergence of `MSR` *in a Cooperative Game.* First, we illustrate convergence of `MSR` in a *cooperative* game on the MNIST and GTSRB datasets. For the MNIST image (index 67) in Figure 9, the black line denotes the descending trend of the upper bound $\mathsf{u}_{\mathtt{MSR}}$, whereas the red line indicates the ascending trend of the lower bound $\mathsf{l}_{\mathtt{MSR}}$. Intuitively, after a few iterations, the upper bound (i.e., minimum

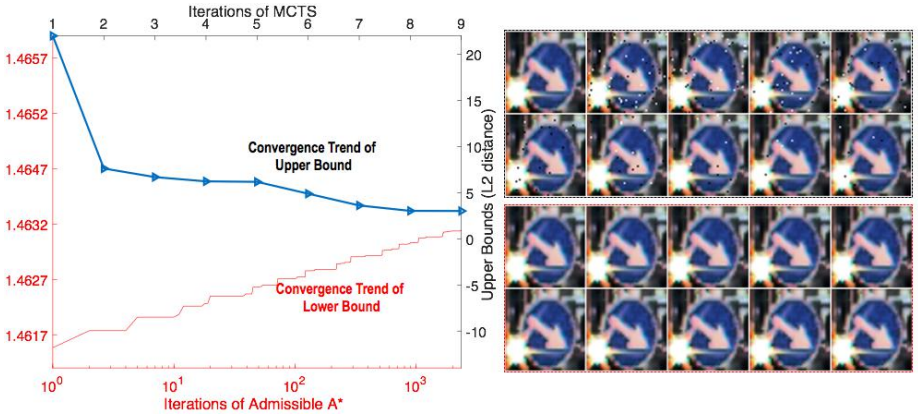


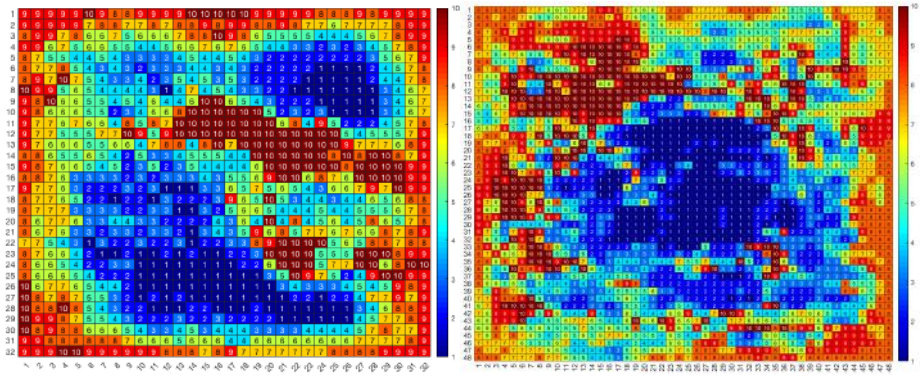
Figure 10: Convergence of *maximum safe radius* in a *cooperative* game with grey-box feature extraction of a GTSRB image originally classified as "keep right". Left: The convergence trends of the upper bound from MCTS and the lower bound from Admissible A* for the MSR problem. Right: The generated *adversarial* images while searching for the upper bound via MCTS, and lower boundary *safe* images while searching for the lower bound via Admissible A*.

distance to an adversarial example) is 2.84 wrt the L_2 metric, and the absolute safety (i.e., lower bound) is within radius 0.012 from the original image. The right-hand side of Figure 9 includes images produced by intermediate iterations, with *adversarial* images generated by MCTS shown in the two top rows, and *safe* images computed by Admissible A* in the bottom rows. Similarly, Figure 10 displays the converging upper and lower bounds of MSR in a *cooperative* game on a GTSRB image (index 19).

As for the computation time, each MCTS iteration updates the upper bound $\mathsf{u}_{\mathtt{MSR}}$ and typically takes minutes; each Admissible A* iteration further expands the game tree and updates the lower bound $\mathsf{l}_{\mathtt{MSR}}$ whenever applicable. The running times for the iterations of the Admissible A* vary: initially it takes minutes but this can increase to hours when the tree is larger.

Convergence of $\mathtt{FR}_\Lambda$ *in a Competitive Game.* Next we demonstrate the convergence of $\mathtt{FR}_\Lambda$ in a *competitive* game on the CIFAR10 and GTSRB datasets. Each iteration of MCTS or Alpha-Beta Pruning updates their respective bound with respect to a certain feature. Note that, in each MCTS iteration, upper bounds $\mathsf{u}_{\mathtt{MSR}}$ of all the features are improved and therefore the maximum among them, i.e., $\mathsf{u}_{\mathtt{FR}_\Lambda}$ of the image, is updated, whereas Alpha-Beta Pruning calculates $\mathsf{l}_{\mathtt{MSR}}$ of a feature in each iteration, and then compares and updates $\mathsf{l}_{\mathtt{FR}_\Lambda}$ with the computation progressing until all the features are processed.

For the CIFAR10 image in Figure 12, the green line denotes the upper bound $\mathsf{u}_{\mathtt{FR}_\Lambda}$ and the red line denotes the lower bound $\mathsf{l}_{\mathtt{FR}_\Lambda}$. The "ship" image is partitioned into 10 features (see Figure 11a) utilising the grey-box extraction method. We observe that this saliency-guided image segmentation procedure captures the features well, as in Figure 11a the most influential features (in blue) resemble the



(a) CIFAR10 "ship" image. (b) GTSRB "speed limit 70" image.

Figure 11: Illustration of the 10 features using the grey-box feature extraction procedure: cells with the same colour indicate the same feature, and the number in each cell represents the featureID. That is, Feature1 in deep blue has the most salient impact, whereas Feature10 in deep red is the least influential. (a) Features of the CIFAR10 "ship" image (32×32) in Figure 12. (b) Features of the GTSRB "speed limit 70" image (48×48) in Figure 13.

silhouette of the "ship". After 3 iterations, the algorithm indicates that, at L_2 distance of more than 1.75, all features are fragile, and if the L_2 distance is 0.48 there exists at least one robust feature. The right-hand side of Figure 12 shows several intermediate images produced, along with the converging $\mathsf{u}_{\mathtt{FR}_\Lambda}$ and $\mathsf{l}_{\mathtt{FR}_\Lambda}$. The top row exhibits the original image as well as the manipulated images with decreasing $\mathsf{u}_{\mathtt{FR}}$. For instance, after the 1st iteration, MCTS finds an adversary perturbed in Feature4 with L_2 distance 2.38, which means by far the most robust feature of this "ship" image is Feature4. (FeatureID is retrieved from the number in each cell of the image segmentation in Figure 11a.) When the computation proceeds, the 2nd iteration updates $\mathsf{u}_{\mathtt{FR}_\Lambda}$ from 2.38 to 1.94, and explores the current most robust Feature8, which is again replaced by Feature9 after the 3rd iteration with lower distance 1.75. The bottom row displays the original image together with perturbations in each feature while $\mathsf{l}_{\mathtt{FR}_\Lambda}$ is increasing. It can be seen that Feature1, Feature2, and Feature3 need only one dimension change to cause image misclassification, and the lower bound Feature4 increases from 0.42 to 0.56 after three iterations.

For the *feature robustness* ($\mathtt{FR}_\Lambda$) problem, i.e., when Player I and Player II are competing against each other, apart from the previous CIFAR10 case where Player II wins the game by generating an adversarial example with atomic manipulations in each feature, there is a chance that Player I wins, i.e., at least one robust feature exists. Figure 13 illustrates this scenario on the GTSRB dataset. Here Player I defeats Player II through finding at least one robust feature by MCTS, and thus the convergence trend of the upper bound $\mathsf{u}_{\mathtt{FR}_\Lambda}$ is not shown. As for the lower bound $\mathsf{l}_{\mathtt{FR}_\Lambda}$, Alpha-Beta Pruning enables Player II to manipulate a single pixel in Feature1 - Feature5 (see Figure 11b) so that adversarial examples are found. For instance, with L_1 distance above 0.79,

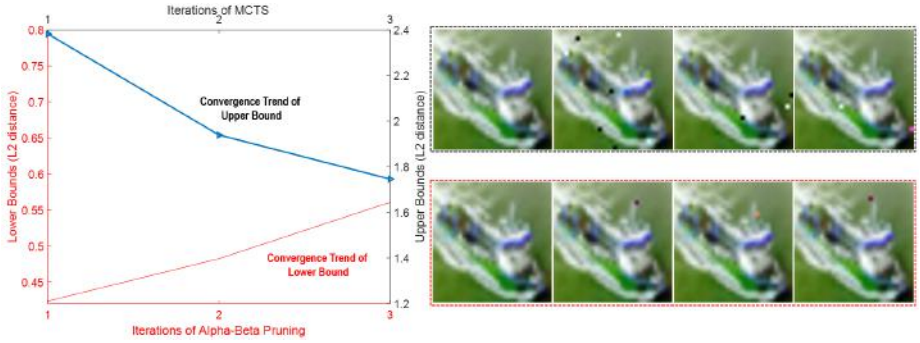


Figure 12: Convergence of *feature robustness* in a *competitive* game with grey-box feature extraction of a CIFAR10 image originally classified as "ship". Left: The convergence trends of the upper bound from MCTS and the lower bound from Alpha-Beta Pruning for the $\mathtt{FR}_\Lambda$ problem. Right: The generated adversarial images while computing the upper bounds via MCTS, and lower bound images while computing the lower bounds via Alpha-Beta Pruning.

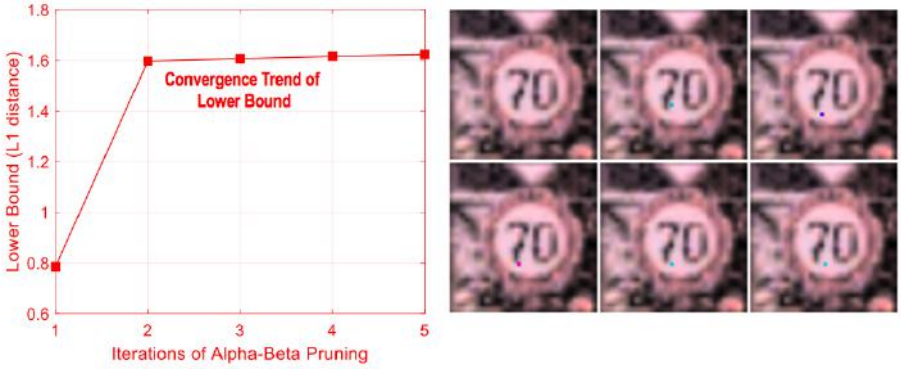


Figure 13: Convergence of *feature robustness* in a *competitive* game with the grey-box feature extraction of a GTSRB image originally classified as "speed limit 70". Left: The convergence trends of the lower bound from Alpha-Beta Pruning for the $\mathtt{FR}_\Lambda$ problem. Right: The generated lower bound images while computing the lower bounds via Alpha-Beta Pruning.

Feature1 turns out to be fragile.

Here, each iteration of MCTS or Alpha-Beta Pruning is dependent on the size of feature partitions – for smaller partitions it takes seconds to minutes, whilst for larger partitions it can take hours. The running times are also dependent on the norm ball radius d. If the radius d is small, the computation can always terminate in minutes.

Scalability wrt Number of Input Dimensions. We now investigate how the increase in the number of dimensions affects the convergence of the lower and upper bounds. From the complexity analysis of the problems in Section 4.3, we know that the theoretical complexity is in PTIME with respect to the size of the game model, which is exponential with respect to the number of input

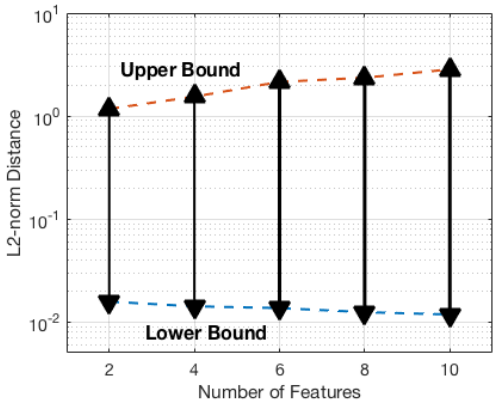


Figure 14: Analysis of convergence of upper and lower bounds of the *maximum safe radius* as the number of dimensions increases for the MNIST image in Figure 9, based on the L_2 norm. The increase in the number of features (2 to 10) corresponds to an increase in the number the input dimensions.

dimensions.

We utilise the example in Figure 9, where the convergence of the upper bound $\mathsf{u}_{\mathtt{MSR}}$ and the lower bound $\mathsf{l}_{\mathtt{MSR}}$ in a cooperative game is exhibited on *all* the dimensions (pixels) of the MNIST image (index 67). We partition the image into 10 disjoint features using the grey-box extraction method, and gradually manipulate features, starting from those with fewer dimensions, to observe how the corresponding bound values $\mathsf{u}_{\mathtt{MSR}}$, $\mathsf{l}_{\mathtt{MSR}}$ are affected if we fix a time budget. To ensure fair comparison, we run the same number of expansions of the game tree, i.e., 10 iterations of MCTS, and 1000 iterations of Admissible A*, and plot the bound values $\mathsf{u}_{\mathtt{MSR}}$, $\mathsf{l}_{\mathtt{MSR}}$ thus obtained. Figure 14 shows the widening upper and lower bounds based on the L_2 norm with respect to 2 to 10 features of the image. It is straightforward to see that the conclusion also holds for the *feature robustness* problem.

6.4. Comparison with Existing Approaches for Generating Adversarial Examples

When the game is cooperative, i.e., for the *maximum safe radius* problem, we can have adversarial examples as by-products. In this regard, both MCTS and A* algorithm can be applied to generate adversarial examples. Note that, for the latter, we can take Inadmissible A* (i.e., the heuristic function can be inadmissible), as the goal is not to ensure the lower bound but to find adversarial examples. By proportionally enlarging the heuristic distance $heuristic(\alpha')$ with a constant, we ask the algorithm to explore those tree nodes where an adversarial example is more likely to be found. Figure 15 displays some adversarial MNIST, CIFAR10 and GTSRB images generated by DeepGame after manipulating a few pixels. More examples can be found in Figures A.17, A.18, and A.19 in the Appendix.

Figure 15: Examples of adversarial MNIST, CIFAR10 and GTSRB images with slight perturbations based on the L_2-norm. Top: "9" misclassified into "8"; "1" misclassified into "3". Middle: "frog" misclassified into "dog"; "dog" misclassified into "cat". Bottom: "speed limit 80" misclassified into "speed limit 60"; "danger" misclassified into "pedestrian crossing".

We compare our tool DeepGame with several state-of-the-art approaches to search for adversarial examples: CW [11], L0-TRE [17], DLV [13], SafeCV [23], and JSMA [10]. More specifically, we train neural networks on two benchmark datasets, MNIST and CIFAR10, and calculate the distance between the adversarial image and the original image based on the L_0-norm. The original images, preprocessed to be within the bound $[0, 1]$, are the first 1000 images of each testing set. Apart from a ten-minute time constraint, we evaluate on correctly classified images and their corresponding adversarial examples. This is because some tools regard misclassified images as adversarial examples and record zero-value distance while other tools do not, which would result in unfair comparison. The hardware environment is a Linux server with NVIDIA GeForce GTX TITAN Black GPUs, and the operating system is Ubuntu 14.04.3 LTS.

Table 1 demonstrates the statistics. Figure A.17 and Figure A.18 in the Appendix include adversarial examples found by these tools. Model architectures, descriptions of the datasets and baseline methods, together with the parameter settings for these tools, can be found in Appendix A.

6.5. Evaluating Safety-Critical Networks

We explore the possibility of applying our game-based approach to support real-time decision making and testing, for which the algorithm needs to be highly efficient, requiring only seconds to execute a task.

We apply our method to a network used for classifying traffic light images collected from dashboard cameras. The Nexar traffic light challenge [22] made over eighteen thousand dashboard camera images publicly available. Each image

[3]Whilst DeepGame works on channel-level dimension of an image, in order to align with some tools that attack at pixel level the statistics are all based on the number of different pixels.

Table 1: Comparison between our tool DeepGame and several other tools on search for adversarial examples performed on the MNIST and CIFAR10 datasets, based on the L0-norm. Here DeepGame deploys the grey-box feature extraction method, and Inadmissible A* algorithm. We set a ten-minute time constraint and evaluate on correctly classified images and the produced adversarial examples.

L_0	MNIST				CIFAR10[3]			
	Distance		Time(s)		Distance		Time(s)	
	mean	std	mean	std	mean	std	mean	std
DeepGame	**6.11**	**2.48**	**4.06**	**1.62**	**2.86**	**1.97**	**5.12**	**3.62**
CW	7.07	4.91	17.06	1.80	3.52	2.67	15.61	5.84
L0-TRE	10.85	6.15	0.17	0.06	2.62	2.55	0.25	0.05
DLV	13.02	5.34	180.79	64.01	3.52	2.23	157.72	21.09
SafeCV	27.96	17.77	12.37	7.71	9.19	9.42	26.31	78.38
JSMA	33.86	22.07	3.16	2.62	19.61	20.94	0.79	1.15

is labelled either green, if the traffic light appearing in the image is green, or red, if the traffic light appearing in the image is red, or null if there is no traffic light appearing in the image. We test the winner of the challenge which scored an accuracy above 90% [35]. Despite each input being 37632-dimensional ($112 \times 112 \times 3$), our algorithm reports that the manipulation of an average of 4.85 dimensions changes the network classification. We illustrate the results of our analysis of the network in Figure 16. Although the images are easy for humans to classify, only one pixel change causes the network to make potentially disastrous decisions, particularly for the case of red light misclassified as green. To explore this particular situation in greater depth, we use a targeted safety MCTS procedure on the same 1000 images, aiming to manipulate images into green. We do not consider images which are already classified as green. Of the remaining 500 images, our algorithm is able to change all image classifications to green with worryingly low distances, namely an average L_0 of 3.23.

7. Related Work

In this section we review works related to safety and robustness verification for neural networks, Lipschitz constant estimation and feature extraction.

7.1. White-box Heuristic Approaches

In [8], Szegedy et. al. find a targeted adversarial example by running the L-BFGS algorithm, which minimises the L_2 distance between the images while maintaining the misclassification. Fast Gradient Sign Method (FGSM) [9], a refinement of L-BFGS, takes as inputs the parameters θ of the model, the input α to the model, and the target label y, and computes a linearized version of the cost function with respect to θ to obtain a manipulation direction. After the

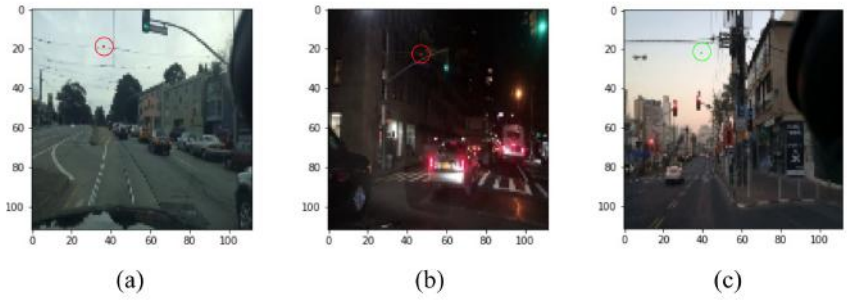


Figure 16: Adversarial examples generated on Nexar data demonstrate a lack of robustness. (a) Green light classified as red with confidence 56% after one pixel change. (b) Green light classified as red with confidence 76% after one pixel change. (c) Red light classified as green with 90% confidence after one pixel change.

manipulation direction is fixed, a small constant value τ is taken as the magnitude of the manipulation. Carlini and Wagner [11] adapt the optimisation problem proposed in [8] to obtain a set of optimisation problems for L_0, L_2, and L_∞ attacks. They claim better performance than FGSM and Jacobian-based Saliency Map Attack (JSMA) with their L_2 attack, in which for every pixel x_i a new real-valued variable w_i is introduced and then the optimisation is conducted by letting x_i move along the gradient direction of $\tanh(w_i)$. Instead of optimisation, JSMA [10] uses a loss function to create a "saliency map" of the image, which indicates the importance of each pixel on the network's decision. A greedy algorithm is used to gradually modify the most important pixels. In [36], an iterative application of an optimisation approach (such as [8]) is conducted on a set of images one by one to get an accumulated manipulation, which is expected to make a number of inputs misclassified. [37] replaces the softmax layer in a deep network with a multiclass SVM and then finds adversarial examples by performing a gradient computation.

7.2. White-box Verification Approaches

Compared to heuristic search for adversarial examples, verification approaches aim to provide guarantees on the safety of DNNs. An early verification approach [38] encodes the entire network as a set of constraints. The constraints can then be solved with a SAT solver. [12] improves on [38] by handling the ReLU activation functions. The Simplex method for linear programming is extended to work with the piecewise linear ReLU functions that cannot be expressed using linear programming. The approach can scale up to networks with 300 ReLU nodes. In recent work [39] the input vector space is partitioned using clustering and then the method of [12] is used to check the individual partitions. DLV [13] uses multi-path search and layer-by-layer refinement to exhaustively explore a finite region of the vector spaces associated with the input layer or

the hidden layers, and scales to work with state-of-the-art networks such as VGG16. [16] shows that most known layers of DNNs are Lipschitz continuous and presents a verification approach based on global optimisation. DiffAI [40] is a method for training robust neural networks based on abstract interpretation, but is unable to calculate the maximum safe radius `MSR`.

7.3. Lipschitz Continuity

The idea of using Lipschitz continuity to provide guarantees on output behaviour of neural networks has been known for some time. Early work [14, 15] focused on small neural networks (few neurons and layers) that only contain differentiable activation functions such as the sigmoid. These works are mainly concerned with the computation of a Lipschitz constant based on strong assumptions that the network has a number of non-zero derivatives and a finite order Taylor series expansion can be found at each vertex. In contrast, our approach assumes knowledge of a (not necessarily tight) Lipschtz constant and focuses on developing verification algorithms for realistically-sized modern networks that use ReLU activation functions, which are non-differentiable.

7.4. Lipschitz Constant Estimation

There has been a resurgence of interest in Lipschitz constant estimation for neural networks. The approaches of FastLin/FastLip [33] and Crown [34] aim to estimate the Lipschitz constant by considering the analytical form of the DNN layers. They are able to compute the bounds, but, in contrast to our approach, which gradually improves the bounds, are not able to improve them. Moreover, their algorithms require access to complete information (e.g., architecture, parameters, etc) about the DNN, while our approach is mainly "black-box" with a (not necessarily tight) Lipschitz constant. We remark that a loose Lipschitz constant can be computed quite easily, noting that a tighter constant can improve computational performance.

Although estimation of the Lipschitz constant has not been the focus of this paper, knowledge of the Lipschitz constant is important in safety verification of DNNs (i.e., estimation of `MSR`). The recent tool DeepGO [16] develops a dynamic Lipschitz constant estimation method for DNNs by taking advantage of advances in Lipschitzian optimisation, through which we can construct both lower and upper bounds for `MSR` with the guarantee of anytime convergence.

7.5. Maximum Safe Radius Computation

Recent approaches to verification for neural networks can also be used to compute bounds on the maximum safety radius directly for, say, L_∞, by gradually enlarging the region. These works can be classified into two categories. The first concentrates on estimation of the lower bound of `MSR` using various techniques. For example, FastLin/FastLip [33] and Crown [34] employ layer-by-layer analysis to obtain a tight lower bound by linearly bounding the ReLU (i.e., FastLin/FastLip) or non-linear activation functions (i.e., Crown). Kolter&Wong

[41, 42], on the other hand, calculates the lower bound of MSR by taking advantage of robust optimisation. The second category aims to adapt abstract interpretation techniques to prove safety. For example, DeepZ and DeepPoly [43] (also including AI^2) adapt abstract interpretation to perform layer-by-layer analysis to over-approximate the outputs for a set of inputs, so that some safety properties can be verified, but are unable to prove absence of safety. A fundamental advantage of DeepGame compared to those works is that it can perform anytime estimation of MSR by improving both lower and upper bounds monotonically, even with a loose Lipschitz constant. Moreover, DeepGame provides a theoretical guarantee that it can reach the exact value of MSR.

Maximal radius computation for DNNs has been addressed directly in [44, 45], where the entire DNN is encoded as a set of constraints, which are then solved by searching for valid solutions to the corresponding satisfiability or optimality problem. The approach of [44] searches for a bound on the maximal safety radius by utilising Reluplex and performing binary search, and [45] instead considers an MILP-based approach. In contrast, our approach utilises a Lipschitz constant to perform search over the input space. Further, our approach only needs to know the Lipschitz constant, whereas [44, 45] need access to the DNN architecture and the trained parameters.

7.6. Black-box Algorithms

The methods in [46] evaluate a network by generating a synthetic data set, training a surrogate model, and then applying white box detection techniques on the model. [47] randomly searches the vector space around the input image for changes which will cause a misclassification. It shows that in some instances this method is efficient and able to indicate where salient areas of the image exist. [17] and this paper are black-box, except that grey-box feature extraction techniques are also considered in this paper to partition the input dimensions. L0-TRE [17] quantifies the global robustness of a DNN, where global robustness is the expectation of the maximum safe radius over a testing dataset, through iteratively generating lower and upper bounds on the network's robustness.

7.7. Feature Extraction Techniques

Feature extraction is an active area of research in machine learning, where the training data are usually sampled from real world problems and high dimensional. Feature extraction techniques reduce the dimensionality of the training data by using a set of features to represent an input sample. In this paper, feature extraction is used not for reducing dimensionality, but rather to partition the input dimensions into a small set of features. Feature extraction methods can be classified into those that are specific to the problem, such as the SIFT [18], SURF [48] and superpixels [32], which are specific to the object detection, and general methods, such as the techniques for computing for every input dimension its significance to the output [28]. The significance values can be visualised as a saliency map, as done in e.g., [10, 17], but can also be utilised as in this paper to partition the input dimensions.

8. Conclusion

In this work, we present a two-player turn-based game framework for the verification of deep neural networks with provable guarantees. We tackle two problems, *maximum safe radius* and *feature robustness*, which essentially correspond to the absolute (pixel-level) and relative (feature-level) safety of a network against adversarial manipulations. Our framework can deploy various feature extraction or image segmentation approaches, including the saliency-guided grey-box mechanism, and the feature-guided black-box procedure. We develop a software tool DeepGame, and demonstrate its applicability on state-of-the-art networks and dataset benchmarks. Our experiments exhibit converging upper and lower bounds, and are competitive compared to existing approaches to search for adversarial examples. Moreover, our framework can be utilised to evaluate robustness of networks in safety-critical applications such as traffic sign recognition in self-driving cars.

Acknowledgements. Kwiatkowska and Ruan are supported by the EPSRC Mobile Autonomy Programme Grant (EP/M019918/1). Huang gratefully acknowledges NVIDIA Corporation for its support with the donation of GPU, and is partially supported by NSFC (NO. 61772232). Wu is supported by the CSC-PAG Oxford Scholarship.

References

[1] G. Dahl, J. W. Stokes, L. Deng, D. Yu, Large-scale malware classification using random projections and neural networks, in: Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing, IEEE SPS, 2013.

[2] J. Ryan, M.-J. Lin, R. Miikkulainen, Intrusion detection with neural networks, in: M. I. Jordan, M. J. Kearns, S. A. Solla (Eds.), Advances in Neural Information Processing Systems 10, Cambridge, MA: MIT Press, 1998, pp. 943–949.

[3] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars, CoRR.

[4] S. Bittel, V. Kaiser, M. Teichmann, M. Thoma, Pixel-wise segmentation of street with neural networks, CoRR.

[5] P. Sermanet, Y. LeCun, Traffic sign recognition with multi-scale convolutional networks, in: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE, 2011, pp. 2809–2813.

[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, F. Roli, Evasion attacks against machine learning at test time, in: ECML/PKDD 2013, 2013, pp. 387–402.

[8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: International Conference on Learning Representations, 2014.

[9] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, CoRR.

[10] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: Security and Privacy (EuroS&P), 2016 IEEE European Symposium on, IEEE, 2016, pp. 372–387.

[11] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: Security and Privacy (SP), 2017 IEEE Symposium on, IEEE, 2017, pp. 39–57.

[12] G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient smt solver for verifying deep neural networks, in: R. Majumdar, V. Kunčak (Eds.), Computer Aided Verification, Springer International Publishing, Cham, 2017, pp. 97–117.

[13] X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks, in: R. Majumdar, V. Kunčak (Eds.), Computer Aided Verification, Springer International Publishing, Cham, 2017, pp. 3–29.

[14] R. R. Zakrzewski, Verification of a trained neural network accuracy, in: IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222), Vol. 3, 2001, pp. 1657–1662 vol.3.

[15] J. Hull, D. Ward, R. R. Zakrzewski, Verification and validation of neural networks for safety-critical applications, in: Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301), Vol. 6, 2002, pp. 4789–4794 vol.6.

[16] W. Ruan, X. Huang, M. Kwiatkowska, Reachability analysis of deep neural networks with provable guarantees, in: Proceedings, International Joint Conference on Artificial Intelligence (IJCAI), 2018.

[17] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, M. Kwiatkowska, Global robustness evaluation of deep neural networks with provable guarantees for the L0 norm, arXiv preprint arXiv:1804.05805.

[18] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[19] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[20] A. Krizhevsky, Learning multiple layers of features from tiny images, Citeseer, 2009.

[21] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural networks 32 (2012) 323–332.

[22] Nexar, Challenge: Using deep learning for traffic light recognition. https://www.getnexar.com/challenge-1.

[23] M. Wicker, X. Huang, M. Kwiatkowska, Feature-guided black-box safety testing of deep neural networks, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2018, pp. 408–426.

[24] Z. Wang, E. P. Simoncelli, A. C. Bovik, Multiscale structural similarity for image quality assessment, in: Signals, Systems and Computers, Conference Record of the Thirty-Seventh Asilomar Conference on, 2003.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2014.

[26] G. Carlsson, T. Ishkhanov, V. de Silva, A. Zomorodian, On the local behavior of spaces of natural images, International Journal of Computer Vision 76 (1) (2008) 1–12.

[27] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, 2016, pp. 1135–1144.

[28] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: NIPS2017, 2017.

[29] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, D. Kroening, Concolic testing for deep neural networks, arXiv preprint arXiv:1805.00089v1.

[30] G. Chaslot, M. Winands, J. Uiterwijk, H. van den Herik, B. Bouzy, Progressive strategies for monte-carlo tree search, New Mathematics and Natural Computation 4 (3) (2008) 343–359.

[31] L. Kocsis, C. Szepesvári, Bandit based monte-carlo planning, in: European Conference on Machine Learning, Springer, 2006, pp. 282–293.

[32] S.-C. Wei, T.-J. Yen, Superpixels generating from the pixel-based k-means clustering., Journal of Multimedia Processing and Technologies 6 (3) (2015) 77–86.

[33] T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. S. Boning, I. S. Dhillon, L. Daniel, Towards fast computation of certified robustness for relu networks, in: ICML, 2018.

[34] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, L. Daniel, Efficient neural network robustness certification with general activation functions, in: NeurIPS, 2018.

[35] A. Burg, Deep Learning Traffic Lights model for Nexar Competition. https://github.com/burgalon/deep-learning-traffic-lights.

[36] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, Universal adversarial perturbations, in: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, 2017, pp. 86–94.

[37] M. Melis, A. Demontis, B. Biggio, G. Brown, G. Fumera, F. Roli, Is deep learning safe for robot vision? adversarial examples against the icub humanoid, CoRR abs/1708.06939.

[38] L. Pulina, A. Tacchella, An abstraction-refinement approach to verification of artificial neural networks, in: International Conference on Computer Aided Verification, Springer, 2010, pp. 243–257.

[39] D. Gopinath, G. Katz, C. S. Pasareanu, C. Barrett, Deepsafe: A data-driven approach for checking adversarial robustness in neural networks, CoRR.

[40] M. Mirman, T. Gehr, M. Vechev, Differentiable abstract interpretation for provably robust neural networks, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, Stockholmsmssan, Stockholm Sweden, 2018, pp. 3578–3586.

[41] J. Z. Kolter, E. Wong, Provable defenses against adversarial examples via the convex outer adversarial polytope, in: ICML, 2018.

[42] E. Wong, F. Schmidt, J. H. Metzen, J. Z. Kolter, Scaling provable adversarial defenses, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31, Curran Associates, Inc., 2018, pp. 8400–8409.

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

[44] N. Carlini, G. Katz, C. W. Barrett, D. L. Dill, Provably minimally-distorted adversarial examples, 2017.

[45] V. Tjeng, K. Xiao, R. Tedrake, Evaluating robustness of neural networks with mixed integer programming, 2017.

[46] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, A. Swami, Practical black-box attacks against machine learning, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ACM, 2017, pp. 506–519.

[47] N. Narodytska, S. Kasiviswanathan, Simple black-box adversarial attacks on deep neural networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 1310–1318.

[48] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. (3) 346–359.

Appendix A. Experimental Setting for Comparison between DeepGame with Existing Works

Appendix A.1. Model Architectures

Table A.2: Architectures of the MNIST, CIFAR-10, and GTSRB models.

Layer Type	MNIST	CIFAR-10/GTSRB
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Convolution + ReLU	$3 \times 3 \times 32$	$3 \times 3 \times 64$
Max Pooling	2×2	2×2
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Convolution + ReLU	$3 \times 3 \times 64$	$3 \times 3 \times 128$
Max Pooling	2×2	2×2
Flatten		
Fully Connected + ReLU	200	256
Dropout	0.5	0.5
Fully Connected + ReLU	200	256
Fully Connected + Softmax	10	10

- Training Accuracy:
  - MNIST (99.99% on 60,000 images)
  - CIFAR-10 (99.83% on 50,000 images)
- Testing Accuracy:
  - MNIST (99.36% on 10,000 images)
  - CIFAR-10 (78.30% on 10,000 images)

Appendix A.2. Datasets

We perform the comparison on two datasets: MNIST and CIFAR-10. They are standard benchmark datasets for adversarial attack of DNNs, and are widely adopted by all these baseline methods.

- MNIST dataset[4]: an image dataset of handwritten digits, which contains a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalised and centred in a fixed-size image.
- CIFAR-10 dataset[5]: an image dataset of 10 mutually exclusive classes, i.e., 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'. It consists of 60,000 32×32 colour images, with 50,000 for training, and 10,000 for testing.

[4] `http://yann.lecun.com/exdb/mnist/`
[5] `https://www.cs.toronto.edu/~kriz/cifar.html`

Appendix A.3. Baseline Methods

We choose a few well-established baseline methods that can perform state-of-the-art L_0 adversarial attacks. Their codes are all available on GitHub.

- CW[6]: a state-of-the-art adversarial attacking method, which models the attacking problem as an unconstrained optimization problem that is solvable by Adam optimizer in TensorFlow.
- L0-TRE[7]: a tensor-based robustness evaluation tool for the L0-norm, and its competitive L0 attack function is compared in this work.
- DLV[8]: an untargeted DNN verification method based on exhaustive search and MCTS.
- SafeCV[9]: a feature-guided safety verification work based on SIFT features, game theory, and MCTS.
- JSMA[10]: a targeted attack based on L_0-norm, so we perform this attack in a sense that the adversarial examples are misclassified into all classes except the correct one.

Appendix A.4. Parameter Setting

MNIST and CIFAR-10 use the same settings, unless separately specified.

- DeepGame
  - gameType = 'cooperative'
  - bound = 'ub'
  - algorithm = 'A*'
  - eta = ('L0', 30)
  - tau = 1
- CW:
  - targeted = False
  - learning_rate = 0.1
  - max_iteration = 100
- L0-TRE:
  - EPSILON = 0.5
  - L0_UPPER_BOUND = 100

[6] https://github.com/carlini/nn_robust_attacks
[7] https://github.com/TrustAI/L0-TRE
[8] https://github.com/TrustAI/DLV
[9] https://github.com/matthewwicker/SafeCV
[10] https://github.com/bethgelab/foolbox

- DLV:
  - mcts_mode = “sift_twoPlayer”
  - startLayer, maxLayer = -1
  - numOfFeatures = 150
  - featureDims = 1
  - MCTS_level_maximal_time = 30
  - MCTS_all_maximal_time = 120
  - MCTS_multi_samples = 5 (MNIST), 3 (CIFAR-10)
- SafeCV:
  - MANIP = max_manip (MNIST), white_manipulation (CIFAR-10)
  - VISIT_CONSTANT = 1
  - backtracking_constant = 1
  - simulation_cutoff = 75 (MNIST), 100 (CIFAR10)
  - small_image = True
- JSMA:
  - bounds = (0, 1)
  - predicts = ‘logits’

Appendix A.5. Platforms

- Hardware Platform:
  - NVIDIA GeForce GTX TITAN Black
  - Intel(R) Core(TM) i5-4690S CPU @ 3.20GHz × 4
- Software Platform:
  - Ubuntu 14.04.3 LTS
  - Fedora 26 (64-bit)
  - Anaconda, PyCharm

Appendix A.6. Adversarial Images

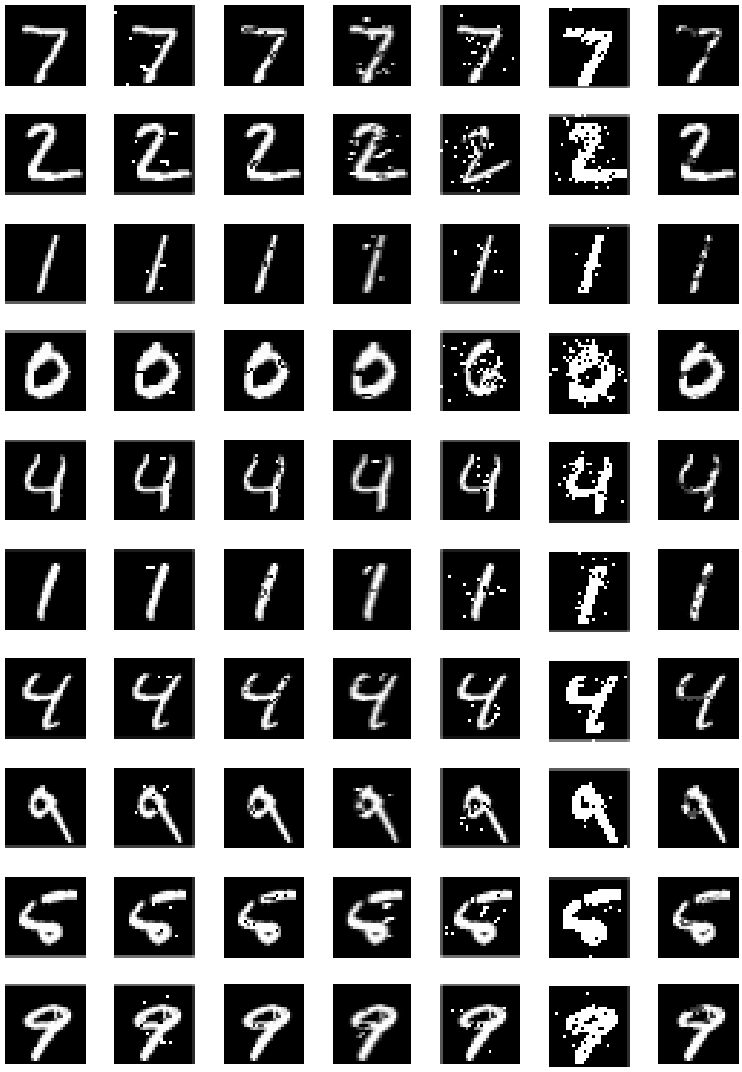


Figure A.17: Comparison of the generated adversarial MNIST images (see Section 6.4). From left to right: original image, DeepGame (this paper), CW, L0-TRE, DLV, SafeCV, and JSMA.

Figure A.18: Comparison of the generated adversarial CIFAR10 images (see Section 6.4. From left to right: original image, DeepGame (this paper), CW, L0-TRE, DLV, SafeCV, and JSMA.

Figure A.19: Adversarial GTSRB images generated by our tool DeepGame.