Feature-Guided Black-Box Safety Testing of Deep Neural Networks

Matthew Wicker[1], Xiaowei Huang[2(✉)], and Marta Kwiatkowska[3]

[1] University of Georgia, Athens, USA
matthew.wicker25@uga.edu

[2] University of Liverpool, Liverpool, UK
xiaowei.huang@liverpool.ac.uk

[3] University of Oxford, Oxford, UK
marta.kwiatkowska@cs.ox.ac.uk

Abstract. Despite the improved accuracy of deep neural networks, the discovery of adversarial examples has raised serious safety concerns. Most existing approaches for crafting adversarial examples necessitate some knowledge (architecture, parameters, etc) of the network at hand. In this paper, we focus on image classifiers and propose a *feature-guided* black-box approach to test the safety of deep neural networks that requires no such knowledge. Our algorithm employs object detection techniques such as SIFT (Scale Invariant Feature Transform) to extract features from an image. These features are converted into a mutable saliency distribution, where high probability is assigned to pixels that affect the composition of the image with respect to the human visual system. We formulate the crafting of adversarial examples as a two-player turn-based stochastic game, where the first player's objective is to minimise the distance to an adversarial example by manipulating the features, and the second player can be cooperative, adversarial, or random. We show that, theoretically, the two-player game can converge to the optimal strategy, and that the optimal strategy represents a globally minimal adversarial image. For Lipschitz networks, we also identify conditions that provide safety guarantees that no adversarial examples exist. Using Monte Carlo tree search we gradually explore the game state space to search for adversarial examples. Our experiments show that, despite the black-box setting, manipulations guided by a perception-based saliency distribution are competitive with state-of-the-art methods that rely on white-box saliency matrices or sophisticated optimization procedures. Finally, we show how our method can be used to evaluate robustness of neural networks in safety-critical applications such as traffic sign recognition in self-driving cars.

1 Introduction

Deep neural networks (DNNs or networks, for simplicity) have been developed for a variety of tasks, including malware detection [11], abnormal network activity detection [31], and self-driving cars [5,6,32]. A classification network N can

D. Beyer and M. Huisman (Eds.): TACAS 2018, LNCS 10805, pp. 408–426, 2018.
https://doi.org/10.1007/978-3-319-89960-2_22

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 [17], they are susceptible to adversarial examples [4,33]. 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 [6], road segmentation [5], and traffic sign classification [32]. 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.

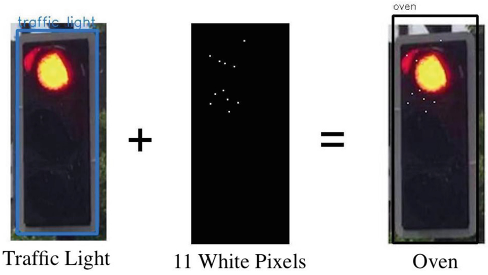


Fig. 1. An adversarial example for the YOLO object recognition network.

A number of approaches have been proposed to search for adversarial examples (see Related Work). They are based on computing the gradients [12], along which a heuristic search moves; computing a Jacobian-based saliency map [27], based on which pixels are selected to be changed; transforming the existence of adversarial examples into an optimisation problem [8], on which an optimisation algorithm can be applied; transforming the existence of adversarial examples into a constraint solving problem [15], 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 [14]. All these approaches assume some knowledge about the network, e.g., the architecture or the parameters, which can vary as the network continuously learns and adapts to new data, and, with a few exceptions [26] that access the penultimate layer, do not explore the feature maps of the networks.

In this paper, we propose a *feature-guided* approach to test the resilience of image classifier networks against adversarial examples. While convolutional neural networks (CNN) have been successful in classification tasks, their feature extraction capability is not well understood [37]. The discovery of adversarial examples has called into question CNN's ability to robustly handle input with diverse structural and compositional elements. On the other hand, state-of-the-art feature extraction methods are able to deterministically and efficiently extract structural elements of an image regardless of scale, rotation or transformation. A key observation of this paper is that feature extraction methods enable us to identify elements of an image which are most vulnerable to a visual system such as a CNN.

Leveraging knowledge of the human perception system, existing object detection techniques detect instances of semantic objects of a certain class (such as animals, buildings, or cars) in digital images and videos by identifying their features. We use the scale-invariant feature transform approach, or SIFT [20], to detect features, which is achieved with no knowledge of the network in a *black-box* manner. Using the SIFT features, whose number is much smaller than the number of pixels, we represent the image as a two-dimensional Gaussian mixture model. This reduction in dimensionality allows us to efficiently target the exploration at salient features, similarly to human perception. We formulate the process of crafting adversarial examples as a two-player turn-based stochastic game, where player `I` selects features and player `II` then selects pixels within the selected features and a manipulation instruction. After both players have made their choices, the image is modified according to the manipulation instruction, and the game continues. While player `I` aims to minimise the distance to an adversarial example, player `II` can be cooperative, adversarial, or nature who samples the pixels according to the Gaussian mixture model. We show that, theoretically, the two-player game can converge to the optimal strategy, and that the optimal strategy represents a globally minimal adversarial image. We also consider safety guarantees for Lipschitz networks and identify conditions to ensure that no adversarial examples exist.

We implement a software package[1], in which a Monte Carlo tree search (MCTS) algorithm is employed to find asymptotically optimal strategies for both players, with player `II` being a cooperator. The algorithm is *anytime*, meaning that it can be terminated with time-out bounds provided by the user and, when terminated, it returns the best strategies it has for both players. The experiments on networks trained on benchmark datasets such as MNIST [18] and CIFAR10 [1] show that, even without the knowledge of the network and using relatively little time (1 min for every image), the algorithm can already achieve competitive performance against existing adversarial example crafting algorithms. We also experiment on several state-of-the-art networks, including the winner of the Nexar traffic light challenge [25], a real-time object detection system YOLO, and VGG16 [3] for ImageNet competition, where, surprisingly, we show that the algorithm can return adversarial examples even with very limited resources (e.g., running time of *less than a second*), including that in Fig. 1 from YOLO. Further, since the SIFT method is scale and rotation invariant, we can counter claims in the recent paper [21] that adversarial examples are not invariant to changes in scale or angle in the physical domain.

Our software package is well suited to safety testing and decision support for DNNs in safety-critical applications. First, the MCTS algorithm can be used *offline* to evaluate the network's robustness against adversarial examples on

[1] The software package and all high-resolution figures used in the paper are available from https://github.com/matthewwicker/SafeCV.

a given set of images. The asymptotic optimal strategy achievable by MCTS algorithm enables a theoretical guarantee of safety, i.e., the network is safe when the algorithm cannot find adversarial examples. The algorithm is guaranteed to terminate, but this may be impractical, so we provide an alternative termination criterion. Second, the MCTS algorithm, in view of its time efficiency, has the potential to be deployed on-board for *real-time* decision support.

An extended version of the paper, which includes more additional explanations and experimental results, is available from [36].

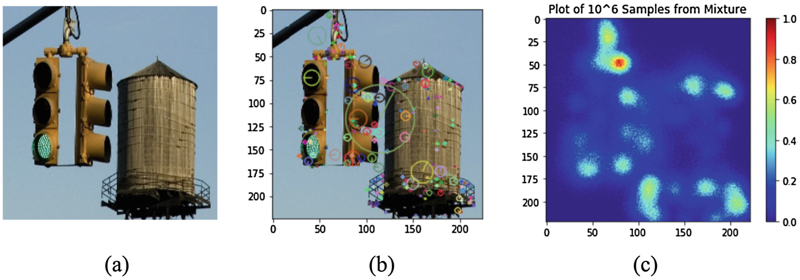
2 Preliminaries

Let N be a 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. For our discussion of image classification networks, the input domain D is a vector space, which in most cases can be represented as $\mathbb{R}^{\mathrm{w} \times \mathrm{h} \times \mathrm{ch}}_{[0,255]}$, where w, h, ch are the width, height, and number of channels of an image, respectively, and we let $P_0 = w \times h \times ch$ be the set of input dimensions. In the following, we may refer to an element in $w \times h$ as a pixel and an element in P_0 as a dimension. We remark that dimensions are normalised as real values in $[0, 1]$. Image classifiers employ a distance function to compare images. Ideally, such a distance should reflect perceptual similarity between images, comparable to human perception. However, in practice L_k distances are used instead, typically L_0, L_1 (Manhattan distance), L_2 (Euclidean distance), and L_∞ (Chebyshev distance). We also work with L_k distances but emphasise that our method can be adapted to other distances. In the following, we write $||\alpha_1 - \alpha_2||_k$ with $k \geq 0$ for the distance between two images α_1 and α_2 with respect to the L_k measurement.

Given an image α, a distance measure L_k, and a distance d, we define $\eta(\alpha, k, d) = \{\alpha' \mid ||\alpha' - \alpha||_k \leq d\}$ as the set of points whose distance to α is no greater than d with respect to L_k. Next we define adversarial examples, as well as what we mean by targeted and non-targeted safety.

Definition 1. *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 $c \neq N(\alpha)$ is such that $\alpha' \in \eta(\alpha, k, d)$, $N(\alpha) \neq N(\alpha')$, and $N(\alpha') = c$. Moreover, we write $adv_{N,k,d}(\alpha, c)$ for the set of adversarial examples of class c and let $adv_{N,k,d}(\alpha) = \bigcup_{c \in C, c \neq N(\alpha)} adv_{N,k,d}(\alpha, c)$. A* targeted safety *of class c is defined as $adv_{N,k,d}(\alpha, c) = \emptyset$, and a* non-targeted safety *is defined as $adv_{N,k,d}(\alpha) = \emptyset$.*

Feature Extraction. The Scale Invariant Feature Transform (SIFT) algorithm [20], a reliable technique for exhuming features from an image, makes object localization and tracking possible without the use of neural networks. Generally, the SIFT algorithm proceeds through the following steps: scale-space extrema detection (detecting relatively darker or lighter areas in the image), keypoint localization (determining the exact position of these areas), and keypoint descriptor assignment (understanding the context of the image w.r.t its local area). Human perception of an image or an object can be reasonably represented as a set of features (referred to as keypoints in SIFT) of different sizes and response strengths, see [35] and Appendix of [36] for more detail. Let $\varLambda(\alpha)$ be a set of features of the image α such that each feature $\lambda \in \varLambda(\alpha)$ is a tuple $(\lambda_x, \lambda_y, \lambda_s, \lambda_r)$, where (λ_x, λ_y) is the coordinate of the feature in the image, λ_s is the size of the feature, and λ_r is the response strength of the feature. The SIFT procedures implemented in standard libraries such as OpenCV may return more information which we do not use.



(a) (b) (c)

Fig. 2. Illustration of the transformation of an image into a saliency distribution. (a) The original image α, provided by ImageNet. (b) The image marked with relevant keypoints $\varLambda(\alpha)$. (c) The heatmap of the Gaussian mixture model $\mathcal{G}(\varLambda(\alpha))$.

On their own, keypoints are not guaranteed to involve every pixel in the image, and in order to ensure a comprehensive and flexible safety analysis, we utilize these keypoints as a basis for a Gaussian mixture model. Figure 2 shows the original image (a) and this image annotated with keypoints (b).

Gaussian Mixture Model. Given an image α and its set $\varLambda(\alpha)$ of keypoints, we define for $\lambda_i \in \varLambda(\alpha)$ a two-dimensional Gaussian distribution $\mathcal{G}_i$ such that, for pixel (p_x, p_y), we have

$$\mathcal{G}_{i,x} = \frac{1}{\sqrt{2\pi\lambda_{i,s}^2}} exp\left(\frac{-(p_x - \lambda_{i,x})^2}{2\lambda_{i,s}^2}\right) \qquad \mathcal{G}_{i,y} = \frac{1}{\sqrt{2\pi\lambda_{i,s}^2}} exp\left(\frac{-(p_y - \lambda_{i,y})^2}{2\lambda_{i,s}^2}\right) \quad (1)$$

where the variance is the size $\lambda_{i,s}$ of the keypoint and the mean is its location $(\lambda_{i,x}, \lambda_{i,y})$. To complete the model, we define a set of weights $\varPhi = \{\phi_i\}_{i \in \{1,2,\ldots,k\}}$ such that $k = |\varLambda(\alpha)|$ and $\phi_i = \lambda_{i,r} / \sum_{j=0}^{k} \lambda_{j,r}$. Then, we can construct a Gaussian mixture model $\mathcal{G}$ by combining the distribution components with the weights as coefficients, i.e., $\mathcal{G}_x = \prod_{i=1}^{k} \phi_i \times \mathcal{G}_{i,x}$ and $\mathcal{G}_y = \prod_{i=1}^{k} \phi_i \times \mathcal{G}_{i,y}$. The two-dimensional distributions are discrete and separable and therefore their realization is tractable and independent, which improves efficiency of computation. Let $\mathcal{G}(\varLambda(\alpha))$ be the obtained Gaussian mixture model from $\varLambda(\alpha)$, and G be the set of

Gaussian mixture models. In Fig. 2 we illustrate the transformation of an image into a saliency distribution.

Pixel Manipulation. We now define the operations that we consider for manipulating images. We write $\alpha(x, y, z)$ for the value of the z-channel (typically RGB or grey-scale values) of the pixel positioned at (x, y) on the image α. Let $I = \{+, -\}$ be a set of manipulation instructions and τ be a positive real number representing the manipulation magnitude, then we can define pixel manipulations $\delta_{X,i} : \mathrm{D} \to \mathrm{D}$ for X a subset of input pixels and $i \in I$:

$$\delta_{X,i}(\alpha)(x, y, z) = \begin{cases} \alpha(x, y, z) + \tau, \text{ if } (x, y) \in X \text{ and } i = + \\ \alpha(x, y, z) - \tau, \text{ if } (x, y) \in X \text{ and } i = - \\ \alpha(x, y, z) \qquad \text{otherwise} \end{cases}$$

for all pixels (x, y) and channels $z \in \{1, 2, 3\}$. Note that if the values are bounded, e.g., $[0, 1]$, $\delta_{X,i}(\alpha)(x, y, z)$ needs to be restricted to be within the bounds. For simplicity, in our experiments and comparisons we allow a manipulation to choose either the upper bound or the lower bound with respect to the instruction i. For example, in Fig. 1, the actual manipulation considered is to make the manipulated dimensions choose value 1.

3 Safety Against Manipulations

Recall that every image represents a point in the input vector space D. Most existing investigations of the safety (or robustness) of DNNs focus on optimising the movement of a point along the gradient direction of some function obtained from the network (see Related Work for more detail). Therefore, these approaches rely on the knowledge about the DNN. Arguably, this reliance holds also for the black-box approach proposed in [26], which uses a new surrogate network trained on the data sampled from the original network. Furthermore, the current understanding about the transferability of adversarial examples (i.e., an adversarial example found for a network can also serve as an adversarial example for another network, trained on different data) are all based on empirical experiments [26]. The conflict between the understanding of transferability and existing approaches to crafting adversarial examples can be gleaned from an observation made in [19] that gradient directions of different models are orthogonal to each other. A reasonable interpretation is that transferable adversarial examples, if they exist, do not rely on the gradient direction suggested by a network but instead may be specific to the input.

In this paper, we propose a *feature-guided* approach which, instead of using the gradient direction as the guide for optimisation, relies on searching fro adversarial examples by targeting and manipulating image features as recognised by human perception capability. We extract features using SIFT, which is a reasonable proxy for human perception and enables dimensionality reduction through the Gaussian mixture representation (see [29]). Our method needs neither the knowledge about the network nor the necessity to massively sample the network for data to train a new network, and is therefore a *black-box* approach.

Game-Based Approach. We formulate the search for adversarial examples as a two-player turn-based stochastic game, where player I selects features and player II then selects pixels within the selected features and a manipulation instruction. While player I aims to minimise the distance to an adversarial example, player II can be cooperative, adversarial, or nature who samples the pixels according to the Gaussian mixture model. To give more intuition for feature-guided search, in Appendix of [36] we demonstrate how the distribution of the Gaussian mixture model representation evolves for different adversarial examples.

We define the objective function in terms of the L_k distance and view the distance to an adversarial example as a measure of its severity. Note that the sets $adv_{N,k,d}(\alpha, c)$ and $adv_{N,k,d}(\alpha)$ of adversarial examples can be infinite.

Definition 2. *Among all adversarial examples in the set $adv_{N,k,d}(\alpha, c)$ (or $adv_{N,k,d}(\alpha)$), find α' with the minimum distance to the original image α:*

$$\arg\min_{\alpha'}\{sev_\alpha(\alpha') \mid \alpha' \in adv_{N,k,d}(\alpha, c)(or\ adv_{N,k,d}(\alpha))\} \tag{2}$$

where $sev_\alpha(\alpha') = ||\alpha - \alpha'||_k$ is the severity *of the adversarial example α' against the original image α.*

We remark that the choice of L_k will affect perceptual similarity, see Appendix of [36].

Crafting Adversarial Examples as a Two-Player Turn-Based Game. Assume two players I and II. Let $M(\alpha, k, d) = (S \cup (S \times \Lambda(\alpha)), s_0, \{T_a\}_{a\in\{\mathtt{I},\mathtt{II}\}}, L)$ be a game model, where S is a set of game states belonging to player I such that each state represents an image in $\eta(\alpha, k, d)$, and $S \times \Lambda(\alpha)$ is a set of game states belonging to player II where $\Lambda(\alpha)$ is a set of features (keypoints) of image α. We write $\alpha(s)$ for the image associated to the state $s \in S$. $s_0 \in S$ is the initial game state such that $\alpha(s_0)$ is the original image α. The transition relation $T_{\mathtt{I}} : S \times \Lambda(\alpha) \to S \times \Lambda(\alpha)$ is defined as $T_{\mathtt{I}}(s, \lambda) = (s, \lambda)$, and transition relation $T_{\mathtt{II}} : (S \times \Lambda(\alpha)) \times \mathcal{P}(P_0) \times I \to S$ is defined as $T_{\mathtt{II}}((s, \lambda), X, i) = \delta_{X,i}(\alpha(s))$, where $\delta_{X,i}$ is a pixel manipulation defined in Sect. 2. Intuitively, on every game state $s \in S$, player I will choose a keypoint λ, and, in response to this, player II will choose a pair (X, i), where X is a set of input dimensions and i is a manipulation instruction. The labelling function $L : S \cup (S \times \Lambda(\alpha)) \to C \times G$ assigns to each state s or (s, λ) a class $N(\alpha(s))$ and a two-dimensional Gaussian mixture model $\mathcal{G}(\Lambda(\alpha(s)))$.

A path (or game play) of the game model is a sequence $s_1u_1s_2u_2...$ 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 $s_{k+1} = T_{\mathtt{II}}((s_k, \lambda_k), X_k, i_k)$ for some (X_k, i_k). Let $last(\rho)$ be the last state of a finite path ρ and $Path^F_a$ be the set of finite paths such that $last(\rho)$ belongs to player $a \in \{\mathtt{I}, \mathtt{II}\}$. A stochastic strategy $\sigma_{\mathtt{I}} : Path^F_{\mathtt{I}} \to \mathcal{D}(\Lambda(\alpha))$ of player I maps each finite paths to a distribution over the next actions, and similarly for $\sigma_{\mathtt{II}} : Path^F_{\mathtt{II}} \to \mathcal{D}(\mathcal{P}(P_0) \times I)$ for player II. We call $\sigma = (\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}})$ a strategy profile. In this section, we only discuss targeted safety for a given target class c (see Definition 1). All the notations and results can be easily adapted to work with non-targeted safety.

In the following, 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 a measure of severity of the adversarial example found over a path. Note that, given σ, the game becomes a fully probabilistic system. Let $\alpha'_\rho = \alpha(last(\rho))$ be the image associated with the last state of the path ρ. We write $t(\rho)$ for the expression $N(\alpha'_\rho) = c \vee ||\alpha'_\rho - \alpha||_k > d$, representing that the path has reached a state whose associated image either is in the target class c or lies outside the region $\eta(\alpha, k, d)$. The path ρ can be terminated whenever $t(\rho)$ is satisfiable. It is not hard to see that, due to the constraints in Definition 1, every infinite path has a finite prefix which can be terminated. Then we define the reward function $R(\sigma, \rho) =$

$$\begin{cases} 1/sev_\alpha(\alpha'_\rho) & \text{if } t(\rho) \text{ and } \rho \in Path_{\mathtt{I}}^F \\ \sum_{\lambda\in\Lambda(\alpha)} \sigma_{\mathtt{I}}(\rho)(\lambda) \cdot R(\sigma, \rho T_{\mathtt{I}}(last(\rho), \lambda)) & \text{if } \neg t(\rho) \text{ and } \rho \in Path_{\mathtt{I}}^F \\ \sum_{(X,i)\in\mathcal{P}(P_0)\times I} \sigma_{\mathtt{II}}(\rho)(X, i) \cdot R(\sigma, \rho T_{\mathtt{II}}(last(\rho), X, i)) & \text{if } \rho \in Path_{\mathtt{II}}^F \end{cases}$$

where $\sigma_{\mathtt{I}}(\rho)(\lambda)$ is the probability of selecting λ on ρ by player I, and $\sigma_{\mathtt{II}}(\rho)(X, i)$ is the probability of selecting (X, i) based on ρ by player II. We note that a path only terminates on player I states.

Intuitively, if an adversarial example is found then the reward assigned is the inverse of severity (minimal distance), and otherwise it is the weighted summation of the rewards if its children. Thus, a strategy $\sigma_{\mathtt{I}}$ to maximise the reward will need to minimise the severity $sev_\alpha(\alpha'_\rho)$, the objective of the problem defined in Definition 2.

Definition 3. *The goal of the game is for player* I *to choose a strategy* $\sigma_{\mathtt{I}}$ *to maximise the reward* $R((\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}}), s_0)$ *of the initial state* s_0*, based on the strategy* $\sigma_{\mathtt{II}}$ *of the player* II*, i.e.,*

$$\arg\max_{\sigma_{\mathtt{I}}} \mathtt{opt}_{\sigma_{\mathtt{II}}} R((\sigma_{\mathtt{I}}, \sigma_{\mathtt{II}}), s_0). \tag{3}$$

where option $\mathtt{opt}_{\sigma_{\mathtt{II}}}$ *can be* $\max_{\sigma_{\mathtt{II}}}$, $\min_{\sigma_{\mathtt{II}}}$, *or* $\mathtt{nat}_{\sigma_{\mathtt{II}}}$, *according to which player* II *acts as a cooperator, an adversary, or nature who samples the distribution* $\mathcal{G}(\Lambda(\alpha))$ *for pixels and randomly chooses the manipulation instruction.*

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

Theorem 1. *Deterministic and memoryless strategies suffice for player* I*, when* $\mathtt{opt}_{\sigma_{\mathtt{II}}} \in \{\max_{\sigma_{\mathtt{II}}}, \min_{\sigma_{\mathtt{II}}}, \mathtt{nat}_{\sigma_{\mathtt{II}}}\}$.

Complexity of the Problem. As a by-product of Theorem 1, the theoretical complexity of the problem (i.e., determining whether $adv_{N,k,d}(\alpha, c) = \emptyset$) is in PTIME, with respect to the size of the game model $M(\alpha, k, d)$. However, even if we only consider finite paths (and therefore a finite system), the number of states (and therefore the size of the system) is $O(|P_0|^h)$ for h the length of the longest finite path of the system without a terminating state. While the precise size of $O(|P_0|^h)$ 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 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, value iteration, and policy iteration, etc) that are used in probabilistic verification.

4 Monte Carlo Tree Search for Asymptotically Optimal Strategy

In this section, we present an approach based on Monte Carlo tree search (MCTS) [9] to find an optimal strategy asymptotically. We also we show that the optimal strategy, if achieved, represents the best adversarial example with respect to the objective in Definition 2, under some conditions.

We first consider the case of $\texttt{opt}_{\sigma_{\text{II}}} = \max_{\sigma_{\text{II}}}$. An MCTS algorithm, whose pseudo-code is presented in Algorithm 1, gradually expands a *partial game tree* by sampling the strategy space of the model $M(\alpha, k, d)$. With the upper confidence bound (UCB) [16] as the exploration-exploitation tradeoff, MCTS has a theoretical guarantee that it converges to optimal solution when the game tree is fully explored. The algorithm mainly follows the standard MCTS procedure, with a few adaptations. We use two termination conditions tc_1 and tc_2 to control the pace of the algorithm. More specifically, tc_1 controls whether the entire procedure should be terminated, and tc_2 controls when a move should be made. The terminating conditions can be, e.g., bounds on the number of iterations, etc. On the partial tree, every node maintains a pair (r, n), which represents the accumulated reward r and the number of visits n, respectively. The *selection* procedure travels from the root to a leaf according to an exploration-exploitation balance, i.e., UCB [16]. After *expanding* the leaf node to have its children added to the partial tree, we call *Simulation* to run simulation on every child node. A simulation on a new *node* is a play of the game from *node* until it terminates. Players act randomly during the simulation. Every simulation terminates when reaching a terminated node α', on which a reward $1/sev_\alpha(\alpha')$ can be computed. This reward is then *backpropagated* from the new child node through its ancestors until reaching the root. Every time a new reward v is backpropogated through a node, we update its associated pair to $(r+v, n+1)$. The $bestChild(root)$ returns the child of *root* which has the highest value of r/n. The other two cases are similar except for the choice of the next move (i.e., Line 12). Instead of choosing the best child, a child is chosen by sampling $\mathcal{G}(\Lambda(\alpha))$ for the case of $\texttt{opt}_{\sigma_{\text{II}}} = \texttt{nat}_{\sigma_{\text{II}}}$, and the worst child is chosen for the case of $\texttt{opt}_{\sigma_{\text{II}}} = \min_{\sigma_{\text{II}}}$. We remark the game is not zero-sum when $\texttt{opt}_{\sigma_{\text{II}}} \in \{\texttt{nat}_{\sigma_{\text{II}}}, \max_{\sigma_{\text{II}}}\}$.

Severity Interval from the Game. Assume that we have fixed termination conditions tc_1 and tc_2 and target class c. Given an option $\texttt{opt}_{\sigma_{\text{II}}}$ for player $\texttt{II}$, we have an MCTS algorithm to compute an adversarial example α'. Let

Algorithm 1. Monte Carlo Tree Search for $\texttt{opt}_{\sigma_{\text{II}}} = \max_{\sigma_{\text{II}}}$

```
Input: A game model $M(\alpha, k, d)$, two termination conditions $tc_1$ and $tc_2$, a target class $c$
Output: An adversarial example $\alpha'$
procedure MCTS$(M(\alpha, k, d), tc_1, tc_2, c)$
    $root \leftarrow s_0$
    While$(\neg tc_1)$:
      While$(\neg tc_2)$:
        $leaf \leftarrow selection(root)$
        $newnodes \leftarrow expansion(M(\alpha, k, d), leaf)$
        for $node$ in $newnodes$:
          $v \leftarrow Simulation(M(\alpha, k, d), node, c)$
          $backPropogation(node, v)$
      $root \leftarrow bestChild(root)$
    return $root$
```

$sev(M(\alpha, k, d), \texttt{opt}_{\sigma_{\text{II}}})$ be $sev_\alpha(\alpha')$, where α' is the returned adversarial example by running Algorithm 1 over the inputs $M(\alpha, k, d)$, tc_1, tc_2, c for a certain $\texttt{opt}_{\sigma_{\text{II}}}$. Then there exists a severity interval $SI(\alpha, k, d)$ with respect to the role of player $\texttt{II}$:

$$[sev(M(\alpha, k, d), \max_{\sigma_{\text{II}}}), \quad sev(M(\alpha, k, d), \min_{\sigma_{\text{II}}})]. \tag{4}$$

Moreover, we have that $sev(M(\alpha, k, d), \texttt{nat}_{\sigma_{\text{II}}}) \in SI(\alpha, k, d)$.

Safety Guarantee via Optimal Strategy. Recall that τ, a positive real number, is the manipulation magnitude used in pixel manipulations. An image $\alpha' \in \eta(\alpha, k, d)$ is a τ-grid image 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 images in $\eta(\alpha, k, d)$. First of all, we have the following conclusion for the case when player $\texttt{II}$ is cooperative.

Theorem 2. *Let $\alpha' \in \eta(\alpha, k, d)$ be any τ-grid image such that $\alpha' \in adv_{N,k,d}(\alpha, c)$, where c is the targeted class. Then we have that $sev_\alpha(\alpha') \geq sev(M(\alpha, k, d), \max_{\sigma_{\text{II}}})$.*

Intuitively, the theorem says that the algorithm can find the optimal adversarial example from the set of τ-grid images. The idea of the proof is to show that every τ-grid image can be reached by some game play. In the following, we show that, if the network is Lipschitz continuous, we need only consider τ-grid images when τ is small enough. Then, together with the above theorem, we can conclude that our algorithm is both sound and complete.

Further, we say that an image $\alpha_1 \in \eta(\alpha, k, d)$ is a misclassification aggregator with respect to a number $\beta > 0$ if, for any $\alpha_2 \in \eta(\alpha_1, 1, \beta)$, 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 input images in $\eta(\alpha_1, 1, \beta)$ are classified correctly. We remark that the region $\eta(\alpha_1, 1, \beta)$ is defined with respect to the L_1

metric, but can also be defined using $L_{k'}$, some k', without affecting the results if $\eta(\alpha, k, d) \subseteq \bigcup_{\alpha_1 \in G(\alpha,k,d)} \eta(\alpha_1, k', \tau/2)$. Then we have the following theorem.

Theorem 3. *If all τ-grid images are misclassification aggregators with respect to $\tau/2$, and $sev(M(\alpha, k, d), \max_{\sigma_{\text{II}}}) > d$, then $adv_{N,k,d}(\alpha, c) = \emptyset$.*

Note that $sev(M(\alpha, k, d), \max_{\sigma_{\text{II}}}) > d$ means that none of the τ-images in $\eta(\alpha, k, d)$ is an adversarial example. The theorem suggests that, to achieve a complete safety verification, one may gradually decrease τ until either $sev(M(\alpha, k, d), \max_{\sigma_{\text{II}}}) \leq d$, in which case we claim the network is unsafe, or the condition that all τ-grid images are misclassification aggregators with respect to $\tau/2$ is satisfiable, in which case we claim the network is safe. In the following, we discuss how to decide the largest τ for a Lipschitz network, in order to satisfy that condition and therefore achieve a complete verification using our approach.

Definition 4. *Network N is a Lipschitz network with respect to the distance L_k and a constant $\hbar > 0$ if, for all $\alpha, \alpha' \in \mathrm{D}$, we have $|N(\alpha', N(\alpha)) - N(\alpha, N(\alpha))| < \hbar \cdot ||\alpha' - \alpha||_k$.*

Note that all networks whose inputs are bounded, including all image classification networks we studied, are Lipschitz networks. Specifically, it is shown in [30] that most known types of layers, including fully-connected, convolutional, ReLU, maxpooling, sigmoid, softmax, etc., are Lipschitz continuous. Moreover, we let ℓ be the minimum confidence gap for a class change, i.e.,

$$\ell = \min\{|N(\alpha', N(\alpha)) - N(\alpha, N(\alpha))| \mid \alpha, \alpha' \in \mathrm{D}, N(\alpha') \neq N(\alpha)\}.$$

The value of ℓ is in $[0, 1]$, dependent on the network, and can be estimated by examining all input examples α' in the training and test data sets, or computed with provable guarantees by reachability analysis [30]. The following theorem can be seen as an instantiation of Theorem 3 by using Lipschitz continuity with $\tau \leq \frac{2\ell}{\hbar}$ to implement the misclassification aggregator.

Theorem 4. *Let N be a Lipschitz network with respect to L_1 and a constant $\hbar$. Then, when $\tau \leq \frac{2\ell}{\hbar}$ and $sev(M(\alpha, k, d), \max_{\sigma_{\text{II}}}) > d$, we have that $adv_{N,k,d}(\alpha, c) = \emptyset$.*

$1/\epsilon$-convergence Because we are working with a finite game, MCTS is guaranteed to converge when the game tree is fully expanded. In the worst case, it may take a very long time to converge. In practice, we can work with $1/\epsilon$-convergence by letting the program terminate when the current best adversarial example has not been improved by finding a less severe one for $\lceil 1/\epsilon \rceil$ iterations, where $\epsilon > 0$ is a small real number.

5 Experimental Results

For our experiments, we let player II be a cooperator, and its move (X, i) is such that for all $(x_1, y_1, z_1), (x_2, y_2, z_2) \in X$ we have $x_1 = x_2$ and $y_1 = y_2$, i.e., one pixel (including 3 dimensions for color images or 1 dimension for grey-scale images) is changed for every move. When running simulations (Line 10 of Algorithm 1), we let $\sigma_{\mathtt{I}}(\lambda) = \lambda_r / \sum_{\lambda \in \Lambda(\alpha)} \lambda_r$ for all keypoints $\lambda \in \Lambda(\alpha)$ and $\mathtt{opt}_{\sigma_{\mathtt{II}}} = \mathtt{nat}_{\sigma_{\mathtt{II}}}$. That is, player I follows a stochastic strategy to choose a keypoint according to its response strength and player II is nature. In this section, we compare our method with existing approaches, show convergence of the MCTS algorithm on limited runs, evaluate safety-critical networks trained on traffic light images, and counter-claim a recent statement regarding adversarial examples in physical domains.

Comparison with Existing Approaches. We compare our approach to two state-of-the-art methods on two image classification networks, trained on the well known benchmark datasets MNIST and CIFAR10. The MNIST image dataset contains images of size 28 × 28 and one channel and the network is trained with the source code given in [2]. The trained network is of medium size with 600,810 real-valued parameters, and achieves state-of-the-art accuracy, exceeding 99%. It has 12 layers, within which there are 2 convolutional layers, as well as layers such as ReLU, dropout, fully-connected layers and a softmax layer. The CIFAR10 dataset contains small images, 32 × 32, with three channels, and the network is trained with the source code from [1] for more than 12 hours. The trained network has 1,250,858 real-valued parameters and includes convolutional layers, ReLU layers, max-pooling layers, dropout layers, fully-connected layers, and a softmax layer. For both networks, the images are preprocessed to make the value of each dimension lie within the bound $[0, 1]$. We randomly select 1000 images $\{\alpha_i\}_{i \in \{1..1000\}}$ from both datasets for non-targeted safety testing. The numbers in Table 1 are average distances defined as $\frac{1}{1000} \cdot \sum_{i=1}^{1000} ||\alpha_i - \alpha_i'||_0$, where α_i' is the adversarial image of α_i returned by the algorithm. Table 1 gives a comparison with the other two approaches (CW [8] and JSMA [27]). The numbers for CW and JSMA are taken from [8][2], where additional optimisations have been conducted over the original JSMA. According to [27], the original JSMA has an average distance of 40 for MNIST.

Table 1. CW vs. Game (this paper) vs. JSMA

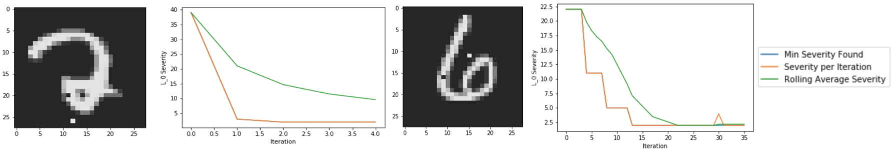
L_0	CW (L_0 algorithm)	Game (timeout = 1 m)	JSMA-F	JSMA-Z
MNIST	8.5	14.1	17	20
CIFAR10	5.8	9	25	20

Our experiments are conducted by setting the termination conditions $tc_1 = 20\,\mathrm{s}$ and $tc_2 = 60\,\mathrm{s}$ for every image. Note that JSMA needs several minutes to

[2] For CW, the L_0 distance in [8] counts the number of changed pixels, while for the others the L_0 distance counts the number of changed dimensions. Therefore, the number 5.8 in Table 1 is not precise, and should be between 5.8 and 17.4, because colour images have three channels.

handle an image, and CW is 10 times slower than JSMA [8]. From the table, we can see that, already in a limited computation time, our game-based approach can achieve a significant margin over optimised JSMA, which is based on saliency distributions, although it is not able to beat the optimisation-based approach CW. We also mention that, in [14], the un-optimised JSMA produces adversarial examples with smaller average L_2 distance than FGSM [12] and DLV on its single-path algorithm [14]. Appendix of [36] provide illustrative examples exhibiting the manipulations that the three algorithms performed on the images.

Convergence in Limited Runs. To demonstrate convergence of our algorithm, we plot the evolution of three variables related to the adversarial severity $sev_\alpha(\alpha')$ against the number of iterations. The variable *best* (in blue color) is the smallest severity found so far. The variable *current* (in orange) is the severity returned in the current iteration. The variable *window* (in green) is the average severity returned in the past 10 iterations. The blue and orange plots may overlap because we let the algorithm return the best example when it fails to find an adversarial example in some iteration. The experiments are terminated with $1/\epsilon$-convergence of different ϵ value such as 0.1 or 0.05. The green plot getting closer to the other two provides empirical evidence of convergence. In Fig. 3 we show that two MNIST images converge over fewer than 50 iterations on manipulations of 2 pixels, and we have confirmed that they represent optimal strategies of the players. We also work with other state-of-the-art networks such as the VGG16 network [3] from the ImageNet competition. Examples of convergence are provided in Appendix of [36].

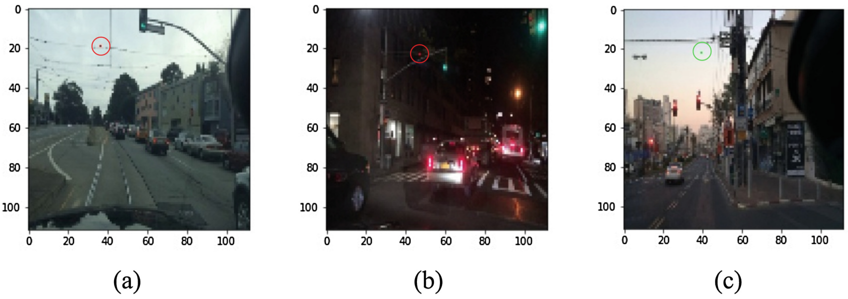


(a) (b) (c) (d)

Fig. 3. (a) Image of a two classified as a seven with 70% confidence and (b) the demonstration of convergence. (c) Image of a six classified as a five with 50% confidence and (d) the demonstration of convergence. (Color figure online)

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 [25] made over eighteen thousand dashboard camera images publicly available. Each image is labeled 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% [7]. 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. Each image was processed by the algorithm in 0.303 s (which includes time to read and write images), i.e., 304 s are taken to test all 1000 images. We illustrate the results of our analysis of the network in Fig. 4. Though 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 severities, namely an average L_0 of 3.23. On average, this targeted procedure returns an adversarial example in 0.21 s per image. Appendix of [36] provides some other examples.



(a)

(b)

(c)

Fig. 4. 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. (Color figure online)

Counter-Claim to Statements in [21]**.** A recent paper [21] argued that, under specific circumstances, there is no need to worry about adversarial examples because they are not invariant to changes in scale or angle in the physical domain. Our SIFT-based approach, which is inherently scale and rotationally invariant, can easily counter-claim such statements. To demonstrate this, we conducted similar tests to [21]. We set up the YOLO network, took pictures of a few traffic lights in Oxford, United Kingdom, and generated adversarial examples on these images. For the adversarial example shown in Fig. 1, we print and photograph it at several different angles and scales to test whether it remains misclassified. The results are shown in Fig. 5. In [21] it is suggested that realistic camera movements – those which change the angle and distance of the viewer – reduce the phenomenon of adversarial examples to a curiosity rather than a safety concern. Here, we show that our adversarial examples, which are predicated on scale and rotationally invariant methods, defeat these claims.

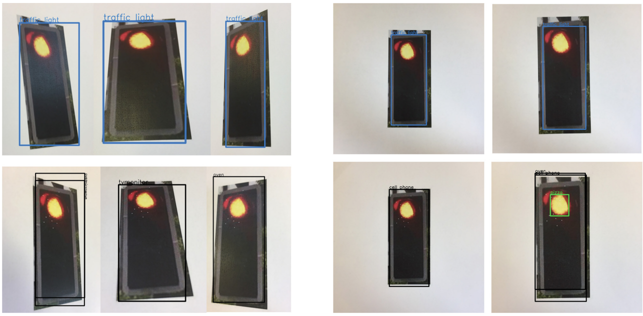


Fig. 5. (Left) Adversarial examples in physical domain remain adversarial at multiple angles. Top images classified correctly as traffic lights, bottom images classified incorrectly as either ovens, TV screens, or microwaves. (Right) Adversarial examples in the physical domain remain adversarial at multiple scales. Top images correctly classified as traffic lights, bottom images classified incorrectly as ovens or microwaves (with the center light being misclassified as a pizza in the bottom right instance).

6 Related Works

We review works concerning the safety (and robustness) of deep neural networks. Instead of trying to be complete, we aim to only cover those directly related.

White-Box Heuristic Approaches. In [34], 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) [12], 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 manipulation direction is fixed, a small constant value τ is taken as the magnitude of the manipulation. Carlini and Wagner [8] adapt the optimisation problem proposed in [34] 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 $\tanh(w_i)$. Different from the optimisation approaches, the JSMA [27] 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 [23], an iterative application of an optimisation approach (such as [34]) 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. [22] replaces the softmax layer in a deep network with a multiclass SVM and then finds adversarial examples by performing a gradient computation.

White-Box Verification Approaches. Compared with heuristic search approaches, the verification approaches aim to provide guarantees on the safety of DNNs. An early verification approach [28] encodes the entire network as a set of constraints. The constraints can then be solved with a SAT solver. [15] improves on [28] 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 [13] the input vector space is partitioned using clustering and then the method of [15] is used to check the individual partitions. DLV [14] 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.

Black-Box Algorithms. The methods in [26] evaluate a network by generating a synthetic data set, training a surrogate model, and then applying white box detection techniques on the model. [24] 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.

7 Conclusion

In this paper we present a novel feature-guided black-box algorithm for evaluating the resilience of deep neural networks against adversarial examples. Our algorithm employs the SIFT method for feature extraction, provides a theoretical safety guarantee under certain restrictions, and is very efficient, opening up the possibility of deployment in real-time decision support. We develop a software package and demonstrate its applicability on a variety of state-of-the-art networks and benchmarks. While we have detected many instabilities in state-of-the-art networks, we have not yet found a network that is safe. Future works include comparison with the Bayesian inference method for identifying adversarial examples [10].

Acknowledgements. Kwiatkowska is supported by EPSRC Mobile Autonomy Programme Grant (EP/M019918/1). Xiaowei gratefully acknowledges NVIDIA Corporation for its support with the donation of the Titan Xp GPU, and is partially supported by NSFC (no. 61772232).

References

1. CIFAR10 model for Keras. https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py
2. MNIST, CNN network. https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py

3. VGG16 model for Keras. https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3
4. Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., Roli, F.: Evasion attacks against machine learning at test time. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 387–402. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_25
5. Bittel, S., Kaiser, V., Teichmann, M., Thoma, M.: Pixel-wise segmentation of street with neural networks. CoRR, abs/1511.00513 (2015)
6. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. CoRR, abs/1604.07316 (2016)
7. Burg, A.: Deep Learning Traffic Lights model for Nexar Competition. https://github.com/burgalon/deep-learning-traffic-lights
8. Carlini, N., Wagner, D.A.: Towards evaluating the robustness of neural networks. CoRR, abs/1608.04644 (2016)
9. Chaslot, G.M.J.B., Winands, M.H.M., Uiterwijk, J.W.H.M., van den Herik, H.J., Bouzy, B.: Progressive strategies for Monte-Carlo tree search. New Math. Nat. Comput. **4**(3), 343–359 (2008)
10. Dabkowski, P., Gal, Y.: Real time image saliency for black box classifiers. CoRR, abs/1705.07857 (2017)
11. Dahl, G., Stokes, J.W., Deng, L., Yu, D.: Large-scale malware classification using random projections and neural networks. In: Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing. IEEE SPS, May 2013
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. CoRR, abs/1412.6572 (2014)
13. Gopinath, D., Katz, G., Pasareanu, C.S., Barrett, C.: Deepsafe: a data-driven approach for checking adversarial robustness in neural networks. CoRR, abs/1710.00486 (2017)
14. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 3–29. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_1
15. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
16. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_29
17. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–444 (2015)
18. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010)
19. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: ICLR 2017 (2017)
20. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
21. Lu, J., Sibai, H., Fabry, E., Forsyth, D.: NO need to worry about adversarial examples in object detection in autonomous vehicles. ArXiv e-prints, July 2017
22. Melis, M., Demontis, A., Biggio, B., Brown, G., Fumera, G., Roli, F.: Is deep learning safe for robot vision? Adversarial examples against the iCub humanoid. CoRR, abs/1708.06939 (2017)

23. Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. CoRR, abs/1610.08401 (2016)
24. Narodytska, N., Kasiviswanathan, S.P.: Simple black-box adversarial perturbations for deep networks. CoRR, abs/1612.06299 (2016)
25. Nexar. Challenge: Using deep learning for traffic light recognition. https://www.getnexar.com/challenge-1
26. Papernot, N., McDaniel, P.D., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against deep learning systems using adversarial examples. CoRR, abs/1602.02697 (2016)
27. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. CoRR, abs/1511.07528 (2015)
28. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 243–257. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_24
29. Reynolds, D.A.: Gaussian mixture models. In: Encyclopedia of Biometrics (2009)
30. Ruan, W., Huang, X., Kwiatkowska, M.: Reachability analysis of deep neural networks with provable guarantees (2018, submitted)
31. Ryan, J., Lin, M.J., Miikkulainen, R.: Intrusion detection with neural networks. In: Jordan, M.I., Kearns, M.J., Solla, S.A. (eds.) Advances in Neural Information Processing Systems, vol. 10, pp. 943–949. MIT Press, Cambridge (1998)
32. Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: The 2011 International Joint Conference on Neural Networks (2011)
33. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR-2014) (2014)
34. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.:. Intriguing properties of neural networks. CoRR, abs/1312.6199 (2013)
35. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, London (2010). https://doi.org/10.1007/978-1-84882-935-0
36. Wicker, M., Huang, X., Kwiatkowska, M.: Feature-guided black-box safety testing of deep neural networks. CoRR, abs/1710.07859 (2017)
37. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. In: 2015 ICML Workshop on Deep Learning (2015)