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

Matthew Wicker[1], Xiaowei Huang[2], and Marta Kwiatkowska[3⋆]

[1] University of Georgia, USA
matthew.wicker25@uga.edu
[2] University of Liverpool, UK
xiaowei.huang@liverpool.ac.uk
[3] University of 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. 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 [12], abnormal network activity detection [34], and self-driving cars [6,5,35]. 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 [20], they are susceptible to adversarial examples [4,36]. An adversarial example is an input which, though initially classified

⋆ Supported by EPSRC Mobile Autonomy Programme Grant.

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 [35]. 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. Fig. 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. Such cases strongly suggest that, before deployment in safety-critical tasks, DNNs resilience (or robustness) to adversarial examples must be strengthened.

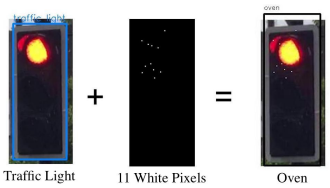


Fig. 1. An adversarial example for the YOLO object recognition network.

A number of approaches have been proposed to search for adversarial examples (see Related Work). They are based on computing the gradients [13], along which a heuristic search moves; computing a Jacobian-based saliency map [31], based on which pixels are selected to 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 [17], 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 [15]. 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 [30] 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 [39]. 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.

Existing object detection techniques, like human perception system, 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 [24], 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, and measure image simi-

larity using with Kullblack-Leibler divergence. 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 also identify conditions on the network and inputs to ensure that no adversarial examples exist.

We implement a software package[4], 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 [21] and CIFAR10 [1] show that, even without the knowledge of the network and using relatively little time (1 minute 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 [29], 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.

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 example on 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.

2 Preliminaries

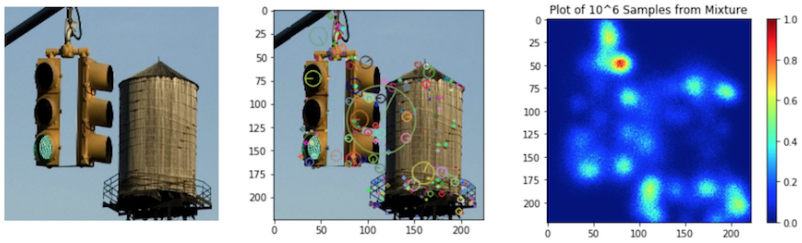
Let N be a network with a set C of classes. Given an input α and a class $c \in C$, we use $N(\alpha, c)$ to denote the confidence (expressed as a probability value obtained from normalising the score) of N believing that α is in class c. Moreover, we write $N(\alpha) = \arg\max_{c \in C} N(\alpha, c)$ 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. The metrics used to compute the distance between images typically include L_0, L_1 (Manhattan distance), L_2 (Euclidean distance), and L_∞

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

(Chebyshev distance). 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.

Definition 1. *(Constraints) Given an input $x \in$ D, a distance measure L_k for some $k \geq 0$, and a distance d, an* adversarial example *α' of class $c \neq N(x)$ 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 [24], 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).



(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 $\Lambda(\alpha)$. (c) The heatmap of the Gaussian mixture model $\mathcal{G}(\Lambda(\alpha))$.

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 [38] and Appendix A for more detail. Let $\Lambda(\alpha)$ be a set of features of the image α such that each feature $\lambda \in \Lambda(\alpha)$ 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. Fig 2 (b) shows an image of (a) annotated with keypoints.

Distance w.r.t. Human Perception Given an image α and its set $\Lambda(\alpha)$ of keypoints, we define for $\lambda_i \in \Lambda(\alpha)$ a two-dimensional Gaussian distribution $\mathcal{G}_i$ such that, for pixel (p_x, p_y), we have

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

where the variance is the size $\lambda_{i,s}$ of the keypoint and the mean is its location $(\lambda_{i,x}, \lambda_{i,y})$. Moreover, we define a set of weights $\Phi = \{\phi_i\}_{i\in\{1,2,\ldots,k\}}$ such that $k = |\Lambda(\alpha)|$ 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 separable, and therefore can be computed independently, which improves efficiency of computation. Let $\mathcal{G}(\Lambda(\alpha))$ be the obtained Gaussian mixture model from $\Lambda(\alpha)$, and G be the set of Gaussian mixture models. In figure 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 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 $X \subseteq P_0$ a subset of input dimensions 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 based on Human Perception

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 [30], which uses JSMA on 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 [30]. The conflict between the understanding of transferability and existing approaches to crafting adversarial examples can be gleaned from an observation made in [22] 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, it relies on 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 [33]). In addition, 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. To give an intuition for feature-guided search, in Fig 15 of Appendix C we demonstrate how the distribution of the Gaussian mixture model representation evolves for different adversarial examples. This key observation – that the network's instability is reflected in its feature maps – justifies the need to consider not only the L_k distance, but also the distance w.r.t. human perception, when formulating an optimisation objective.

Objective to Optimise The sets $adv_{N,k,d}(\alpha, c)$ and $adv_{N,k,d}(\alpha)$ can be infinite. Therefore, we may be interested in finding a finite subset (or singleton) by optimising over an objective function. We derive a suitable objective by incorporating the distance w.r.t. human perception within the usual objective based on the L_k metric.

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

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

where $sev_\alpha(\alpha') = ||\alpha - \alpha'||_k + e * ||\alpha - \alpha'||_H$ is the severity of the adversarial example α' against the original image α and e is an adjustable constant.

We note that existing approaches consider a simpler objective by letting $e = 0$.

Crafting Adversarial Examples as a Two-Player Turn-Based Game Assume two players $\mathtt{I}$ and $\mathtt{II}$. Let $M(\alpha, p, d) = (S \cup (S \times \Lambda(\alpha)), s_0, \{T_a\}_{a\in\{\mathtt{I},\mathtt{II}\}}, L)$ be a game model, where S is a set of game states belonging to player $\mathtt{I}$ 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 $\mathtt{II}$ 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 Section 2. Intuitively, on every game state $s \in S$, player $\mathtt{I}$ will choose a keypoint λ, and, in response to this, player $\mathtt{II}$ 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 $s_1 u_1 s_2 u_2 ...$ 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 $s_{k+1} = T_\mathtt{II}((s_k, \lambda_k), X_k, i_k)$ 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 $\mathtt{I}$ 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 $\mathtt{II}$. We call $\sigma = (\sigma_\mathtt{I}, \sigma_\mathtt{II})$ a strategy profile. In this section, we only discuss targeted safety for the target class c. 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 $\rho \in \bigcup_{a \in \{\mathtt{I},\mathtt{II}\}} Path_a^F$. 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 $\mathtt{I}$, and $\sigma_{\mathtt{II}}(\rho)(X, i)$ is the probability of selecting (X, i) based on ρ by player $\mathtt{II}$. We note that a path only terminates on player $\mathtt{I}$ states.

Definition 3. *The goal of the game is for player* $\mathtt{I}$ *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* $\mathtt{II}$*, 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* $\mathtt{II}$ *acts as a cooperator, an adversary, or nature who samples the distribution* $\mathcal{G}(\Lambda(\alpha))$ *for pixels and randomly chooses the manipulation instruction.*

It is noted that 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. 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. We have the following result.

Theorem 1. *Deterministic and memoryless strategies suffice for player* $\mathtt{I}$*, 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 $M(\alpha, p, d)$. 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 [9] (MCTS) 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 $\texttt{opt}_{\sigma_{\texttt{II}}} = \max_{\sigma_{\texttt{II}}}$. 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 $M(\alpha, p, d)$. With the upper confidence bound (UCB) [18] 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 or the number of iterations, etc. On the partial tree, every node maintains a pair (r, n), which represents the accumulated reward and the number of visits, respectively. The $selection$ travels from the root to a leaf according to an exploration-exploitation balance, i.e., UCB [18]. After $expanding$ the children of the leaf node, we call $Simulation$ to run simulation on every child node. 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 $root$ who has the highest value of the expression r/n.

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

```
1:  Input: A game model M(α, p, d), two termination conditions tc_1 and tc_2, a target class c
2:  Output: An adversarial example α'
3:  procedure MCTS(M(α, p, d), tc_1, tc_2, c)
4:      root ← s_0
5:      While(¬tc_1):
6:          While(¬tc_2):
7:              leaf ← selection(root)
8:              newnodes ← expansion(M(α, p, d), leaf)
9:              for node in newnodes:
10:                 v ← Simulation(M(α, p, d), node, c)
11:                 backPropogation(node, v)
12:         root ← bestChild(root)
13:     return root
```

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 $\texttt{opt}_{\sigma_{\texttt{II}}} = \texttt{nat}_{\sigma_{\texttt{II}}}$, and the worst child is chosen for the case of $\texttt{opt}_{\sigma_{\texttt{II}}} = \min_{\sigma_{\texttt{II}}}$.

Severity Interval from the Game Given an option $\texttt{opt}_{\sigma_{\texttt{II}}}$ for player II, we have an MCTS algorithm to compute an adversarial example α'. Let $sev(M(\alpha, p, d), \texttt{opt}_{\sigma_{\texttt{II}}}) = sev_\alpha(\alpha')$ be the severity of adversarial examples for $M(\alpha, p, d)$ and $\texttt{opt}_{\sigma_{\texttt{II}}}$. Then there

exists a severity interval $SI(\alpha, p, d)$ with respect to the role of player II:

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

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

Safety Guarantee via Optimal Strategy Recall that τ is the manipulation magnitude used in pixel manipulation. 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 $\tau(\alpha, k, d)$ 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 II is cooperative.

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

The idea of the proof is to show that every τ-grid image can be reached by some game play. 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 $N(\alpha_1) \neq N(\alpha)$. Then, we have the following theorem.

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

Proof. (Sketch) First, we can show that $\eta(\alpha, k, d) \subseteq \bigcup_{\alpha_1 \in \tau(\alpha,k,d)} \eta(\alpha_1, 1, \tau/2)$. Now assume that $adv_{N,k,d}(\alpha, c) \neq \emptyset$. Then there must exist an image α' such that $\alpha' \in adv_{N,k,d}(\alpha, c)$. Because all τ-grid images are misclassification aggregators with respect to $\tau/2$, there must exist a τ-grid image α'' such that $\alpha'' \in adv_{N,k,d}(\alpha, c)$. By Theorem 2, we have $sev_\alpha(\alpha'') \geq sev(M(\alpha, p, d, \tau), max_{\sigma_{\text{II}}})$. By the hypothesis that $sev(M(\alpha, p, d), max_{\sigma_{\text{II}}}) > d$, we have $sev_\alpha(\alpha'') > d$, which is impossible because $\alpha'' \in adv_{N,k,d}(\alpha, c) \subset \eta(\alpha, k, d)$. □

The theorem suggests that, to achieve a complete safety verification, one may gradually decrease τ until either $sev(M(\alpha, p, d), max_{\sigma_{\text{II}}}) \leq d$, 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 network satisfying a Lipschitz condition, 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 measure 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. 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 over-estimated, e.g., for MNIST it is roughly 0.5 and for VGG16 is is roughly 0.05. Then we have the following conclusion which can be used to compute the largest τ.

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 $sev(M(\alpha, p, d), max_{\sigma_{\mathtt{II}}}) > d$, we have that $adv_{N,k,d}(\alpha, c) = \emptyset$.*

Proof. We need to show that $\tau \leq \frac{2\ell}{\hbar}$ implies that all τ-grid images are misclassification aggregators with respect to $\tau/2$. First of all, by the definition of Lipschitz network, we have $|N(\alpha_2, N(\alpha_2)) - N(\alpha_1, N(\alpha_2))| < \hbar \cdot ||\alpha_2 - \alpha_1||_1$. Then by the definition of ℓ, we have $||\alpha_2 - \alpha_1||_1 > \ell/h$ when $N(\alpha_2) \neq N(\alpha_1)$. Second, we notice that, the statement that all τ-grid images are misclassification aggregators with respect to $\tau/2$ is equivalent to say that for any τ-grid image α_1 such that $N(\alpha_1) = N(\alpha)$, we have that for any α_2, $N(\alpha_2) \neq N(\alpha_1)$ implies that $||\alpha_2 - \alpha_1||_1 > \tau/2$. Finally, we notice that $||\alpha_2 - \alpha_1||_1 > \tau/2$ holds when $||\alpha_2 - \alpha_1||_1 > \ell/h$ and $\tau \leq \frac{2\ell}{\hbar}$. □

$1/\epsilon$-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 acheives 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. For a fair comparison, we do not use

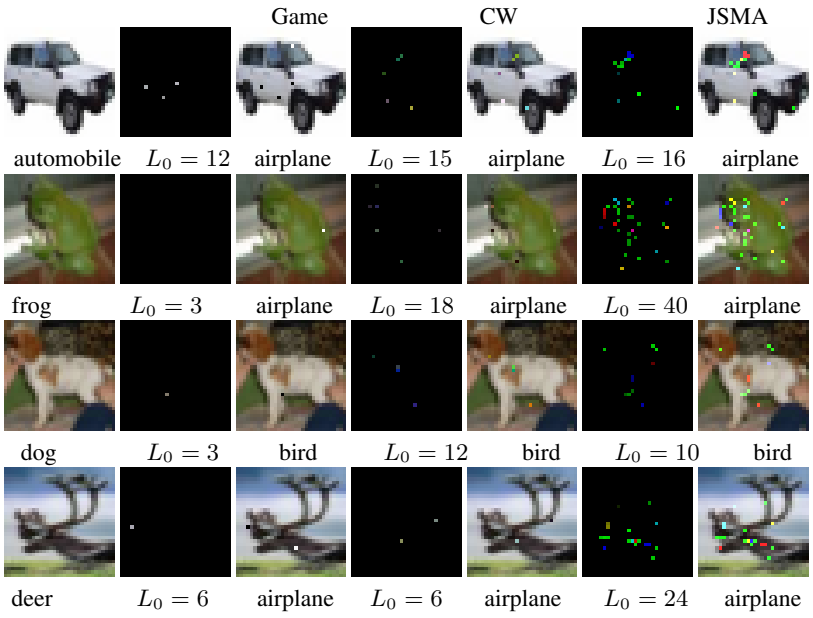


Fig. 3. Adversarial examples by Game (this paper) vs. CW vs. JSMA for CIFAR-10 networks.

the additional D_{KL} distance. Table 1 gives a comparison with the other two approaches (CW [8] and JSMA [31]). The numbers for CW and JSMA are taken from [8] [5], where additional optimisations have been conducted over the original JSMA. According to [31], the original JSMA has an average distance of 40 for MNIST.

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

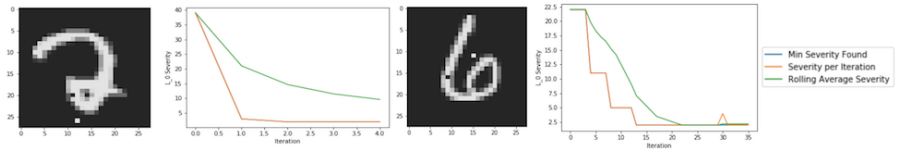
Table 1. CW vs. Game vs. JSMA

Our experiments are conducted by setting the termination conditions $tc_1 = 20s$ and $tc_2 = 60s$ for every image. Note that JSMA needs several minutes to 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 [15], the un-optimised JSMA produces adversarial examples with smaller average L_2 distance than FGSM [13] and DLV on its single-path algorithm [15]. Fig. 3 illustrates 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.

[5] 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.

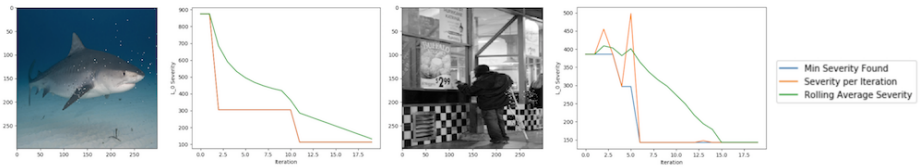
The variable $window$ (in green) is the average severity returned in the past 10 iterations. The blue and orange plots may overlap because we let the algorithm return the best example when it fails to find an adversarial example in some iteration. The experiments are terminated with $1/\epsilon$-convergence of different ϵ value such as 0.1 or 0.05. The convergence of the green plot to the other two provides empirical evidence of c witneencess



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

Fig. 4. (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.

In Fig. 4 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, where plots in Figure 5 show clear convergence. More examples of convergence are provided in Appendix B and later in this section when we evaluate safety-critical applications.



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

Fig. 5. Adversarial examples generated on the VGG16 architecture trained on ImageNet data. (a) Image of a great white shark classified as a galeocerdo cuvieri with confidence 42% after 113 manipulations and (b) the demonstration of convergence over 20 simulations. (b) An image of a crutch classified as bakery after 143 manipulations and (d) the demonstration of convergence over 20 simulations.

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 [29] 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 (112x112x3), 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 seconds (which includes time to read and write images), i.e., 304 seconds are taken to test all 1000 images.

Fig. 6. Adversarial examples generated on Nexar data demonstrate a lack of robustness. (a) Green light classified as red with confidence 56% after one pixel manipulation. (b) Green light classified as red with confidence 76% after one pixel. (c) Red light classified as green with 90% confidence after one pixel.

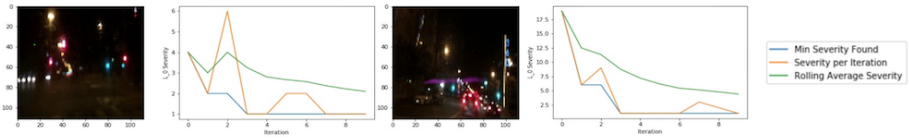
We illustrate the results of our analysis of the network in Fig. 6. Though the images are easy for humans to classify, only one pixel change causes the network to make potentially disastrous decisions, particularly for the case of red light misclassified as green. To explore this particular situation in greater depth, we use a targeted 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 second per image. Some of the more concerning examples are shown in Fig. 7.

We also show in Fig. 8 that, for many inputs, MCTS is able to find an optimal strategy (a single-pixel misclassification) in only eight simulations (about 0.3 seconds).

Counter-claim to Statements in [25] A recent paper [25] 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-approach, which is inherently scale and rotationally invariant, can easily counter-claim such statements. To demonstrate this, we conducted similar tests to [25]. 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. 9. In [25] 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.

(a) (b) (c)

Fig. 7. Targeted adversarial examples on Nexar illustrate safety concerns. (a) Red light classified as green with 68% confidence after one pixel change. (b) Red light classified as green with 95% confidence after one pixel. (c) Red light classified as green with confidence 78% after one pixel.



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

Fig. 8. Convergence to an optimal strategy on Nexar traffic light images. (a) An image of a red light manipulated into a green light after a single pixel change and the plot of convergence over eight simulations (b). (c) An image of a green light manipulated to a red light after a single pixel manipulation and (d) its convergence plot over eight simulations.

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. The first known algorithm for crafting adversarial examples is proposed in [37], where 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. A refinement of the L-BFGS algorithm was given by Goodfellow et al. [13] and named Fast Gradient Sign Method (FGSM). FGSM 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 [37] to obtain a set of optimisation problems for L_0, L_2, and L_∞ attacks. They claim better performance than FGSM and 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 Jacobian-based Saliency Map Attack (JSMA) [31] uses a loss function to create a "saliency map" of the image. This saliency map indicates the importance of each individual pixel on the networks decision. A greedy algorithm is used to gradually modify the most important pixels. In

Fig. 9. (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).

[27], an iterative application of an optimisation approach (such as [37]) 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. [26] 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 approaches, the verification approaches aim to provide guarantees on the safety of DNNs. An early verification approach [32] encodes the entire network as a set of constraints. The constraints can then be solved with a SAT solver. [17] improves on [32] 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 [14] the input vector space is partitioned using clustering and then the method of [17] is used to check the individual partitions. DLV [15] 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 [30] evaluate a network under investigation by generating a synthetic data set, training a surrogate model, and then applying white box detection techniques on the model. Moreover, [28] 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.

Generative adversarial networks (GAN). is a learning algorithm where two neural networks, called discriminator and generator, improve their own capabilities by playing a zero-sum game, in which the role of the generator is to generate inputs as close as possible to the true data and the discriminator identifies the generated fake inputs. Our two-player game is different: it is not used for learning and the players aim to find adversarial examples instead of generating and identifying fake data.

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 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. There are a number of possible future directions, including developing a complete verification approach for non-Lipschitz networks (i.e., the inputs are not bounded) and comparison with the Bayesian inference method for identifying adversarial examples [11].

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. Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML/PKDD 2013*, pages 387–402, 2013.
5. Sebastian Bittel, Vitali Kaiser, Marvin Teichmann, and Martin Thoma. Pixel-wise segmentation of street with neural networks. *CoRR*, abs/1511.00513, 2015.
6. Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.
7. Alon Burg. Deep learning traffic lights, 2017.
8. Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.
9. G.M.J.B. Chaslot, M.H.M. Winands, J.W.H.M. Uiterwijk, H.J. van den Herik, and B. Bouzy. Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation*, 4(3):343–359, 2008.
10. François Chollet. Keras, 2017.
11. Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *CoRR*, abs/1705.07857, 2017.
12. George Dahl, Jack W. Stokes, Li Deng, and Dong Yu. Large-scale malware classification using random projections and neural networks. In *Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing*. IEEE SPS, May 2013.
13. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.
14. Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark Barrett. Deepsafe: A data-driven approach for checking adversarial robustness in neural networks. *CoRR*, abs/1710.00486, 2017.
15. Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *CAV 2017*, pages 3–29, 2017.
16. Itseez. Open source computer vision library, 2015.
17. Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV 2017, to appear*, 2017.
18. Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European Conference on Machine Learning (ECML2006)*, 2006.

19. Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5):363–370, Aug 1984.
20. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.
21. Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.
22. Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR2017*, 2017.
23. David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.
24. David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.
25. J. Lu, H. Sibai, E. Fabry, and D. Forsyth. NO Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles. *ArXiv e-prints*, July 2017.
26. Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. Is deep learning safe for robot vision? adversarial examples against the icub humanoid. *CoRR*, abs/1708.06939, 2017.
27. Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016.
28. Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *CoRR*, abs/1612.06299, 2016.
29. Nexar. Nexar challenge 1 data set, 2017.
30. Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *CoRR*, abs/1602.02697, 2016.
31. Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.
32. Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In *CAV 2010*, pages 243–257, 2010.
33. Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009.
34. Jake Ryan, Meng-Jang Lin, and Risto Miikkulainen. Intrusion detection with neural networks. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 943–949. Cambridge, MA: MIT Press, 1998.
35. Pierre Sermanet and Yann Lecun. Traffic sign recognition with multi-scale convolutional networks. *The 2011 International Joint Conference on Neural Networks*, 2011.
36. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR-2014)*, 2014.
37. Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.
38. Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.
39. Jason Yosinski, Jeff Clune, Thomas Fuchs Anh Nguyen, , and Hod Lipson. Understanding neural networks through deep visualization. In *2015 ICML workshop on Deep learning*, 2015.

A Feature Detection Techniques

In this section, we give a brief review of a state-of-the-art computer vision algorithm which will be used in our black-box approach. The Scale Invariant Feature Transform (SIFT) algorithm [24], 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 in a few 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). Below, we give a summary of the SIFT algorithm, and focus on the output and features which will be used in our algorithms.

Scale-Space Extrema Detection In [19], it is shown that the only appropriate way to parameterize the resolution of an image without the generation of spurious details (i.e. details which are not inherent in the image, but generated by the method of parameterization) is given by the two dimension Gaussian kernel. Lowe [24] uses this to detect extrema in a given image α by observing the local pixel-value extrema at different scales. Formally, the k^{th} scale of an image α is calculated:[6]

$$S(x, y, k\sigma) = G(x, y, k\sigma) * \alpha(x, y) \tag{5}$$

where x, y represent the Euclidean coordinates of a pixel, * is the convolution operator, and $G(x, y, \sigma)$ is the two-dimensional Gaussian kernel given by:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} exp(-(x^2 + y^2)/2\sigma^2) \tag{6}$$

Essentially this parametrization allows us to change σ – the variance of the distribution – in order to achieve different scales. In practice, it has been noted that with this parameterization we are able to filter out some noise of the image, and are able to detect extrema of varying sizes. To get both large and small-sized extrema, we observe the image at a range of scales. Each of the scale ranges is then called an octave, and, after an octave has been calculated, we down-sample the image by a factor of two and observe another octave. In the left images of figure 10, we show the result of applying the Gaussian kernel to a traffic light. It is clear that this blurring removes some of the unnecessary details within the image and leaves some of the larger features to be examined. After the calculation of a scale space range for each octave, Lowe detects extrema by observing the neighbors of a three by three kernel. If a pixel value is larger or smaller than its neighbors in successive scales, then it is marked as a "keypoint" (as shown in the right portion of figure 10).

Importantly, detection of extrema by this method has been shown to be invariant to changes in translation, scaling, rotation, and is minimally affected by noise and small distortions [23]. For our algorithm this means that we should be able to detect and manipulate salient features of images even when the image is of low quality.

[6] The SIFT algorithm uses difference of Gaussians at different scales (i.e $S(x, y, k\sigma) - S(x, y, \sigma)$); for more information see previous work by Lowe [23].

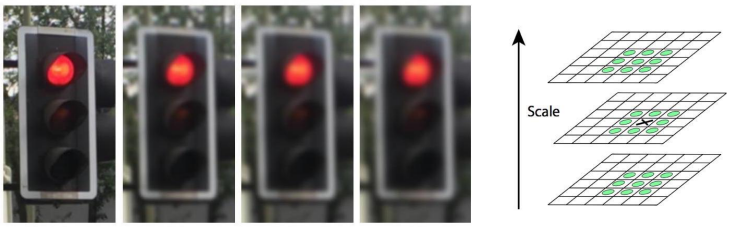


Fig. 10. Demonstration of Gaussian blur affect to generate scale. Far right: figure from [24] shows how extrema are detected from these scaled images. Black 'x' marks an extremum if it is larger or smaller than all of the pixels around it.

Keypoint Description After scale space extrema have been detected, they are located in the original image. Initially, in [24], this localization was done by translating the pixel location from the scale and octave onto the original image; however, this was later improved by using the Taylor expansion of the scale space function shifted so that the origin is at the sampled point. Regardless of which of these is used, the first step of keypoint description is to assign the exact x and y coordinates of the extrema in the image. Once the extrema have been described with a location we refer to them as keypoints in a set Λ where each keypoint $\lambda \in \Lambda$ has an x and y coordinate, λ_x and λ_y, respectively.

After localizing these keypoints, we describe their size and orientation. Size is calculated by the magnitude of the gradient vector corresponding to the keypoint which was located; we will denote the size as λ_s. After size has been calculated, we sample pixel values from different areas around the keypoint to generate descriptors. The implementation of SIFT in [16] (used by our algorithms) gives 128 different local descriptors for each keypoint which includes size, response strength, orientation angle, and local gradient magnitudes. The response strength of keypoints, λ_r, which is derived from the persistence of a keypoint across multiple octaves and scales, and is important for our formulation of a salience distribution.

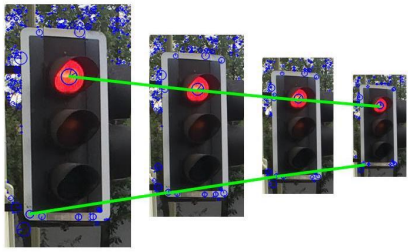


Fig. 11. Locating, describing and matching keypoints across the image. Blue circles are keypoints, while green lines represent a few selected keypoints which are persistent throughout each image size.

B Further Empirical Evidence of MCTS Strategy Convergence

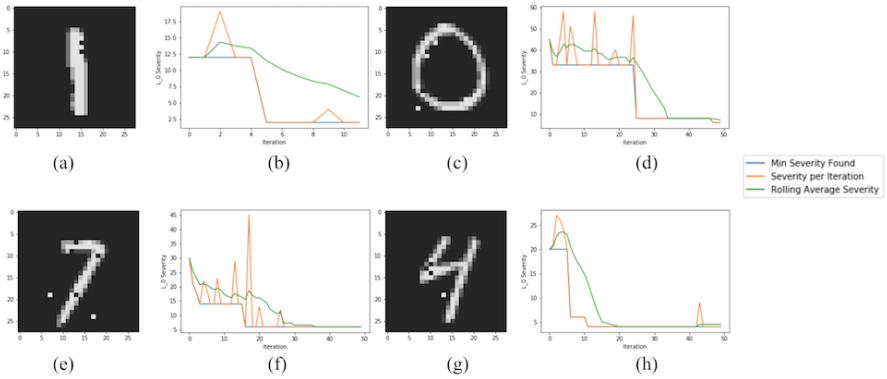


Fig. 12. Further empirical evidence of MCTS strategy convergence. (a) Another optimal convergence example, one modified to an eight with confidence 55%, and (b) plot of MCTS performance over 50 simulations (minimum severity, rolling average severity and severity per iteration). (c) An image of a zero classified as a five with confidence 48% after six pixel manipulations and (d) MCTS performance on this image. (e) Image of a seven classified as an eight with 47% confidence after six pixel manipulations and (f) MCTS performance on this image. (g) Image of a four predicted as an eight with 50% confidence after four pixel manipulations and (h) MCTS performance on this image.

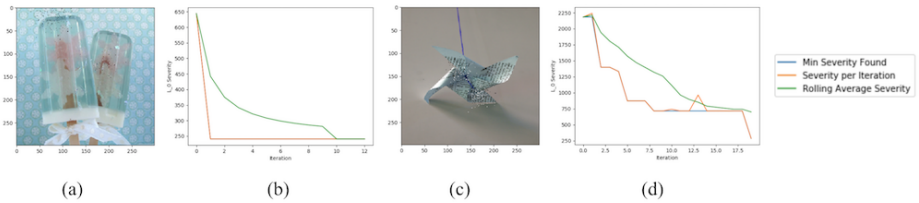


Fig. 13. Further empirical evidence of MCTS strategy convergence on state-of-the-art VGG16 network. (a) Image of an ice lolipop predicted as a nipple with 30% confidence after 241 pixel manipulations and (b) MCTS performance on this image. (c) A pinwheel predicted as a radio telescope with confidence 21% after 287 pixel manipulations and (d) MCTS performance on this image.

C Intuition for Using Feature Detection for Safety Testing

It is reasonable to assume that, if any visual system mistakes the classification of an object, then both the spatial and compositional elements of the image must have played a large role. In artificial visual systems, this mapping between an image's basic elements and its classification is systematically learned; however, the ability to know if a system has truly understood the relation between an image's composition or structure and its classification is difficult, and the advent of adversarial examples suggests that artificial visual systems are very sensitive to perturbations of these elements.

To test this hypothesis – that an artificial visual system is very sensitive to changes in structural or compositional elements – we need to be able to pinpoint and manipulate the most important aspects of such elements.

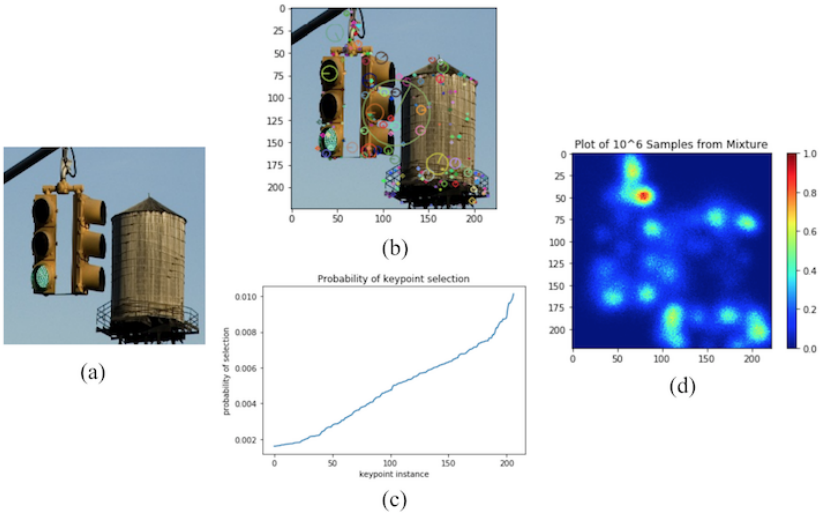


Fig. 14. Illustration of the transformation of an image into a saliency distribution. (a) The original image, provided by ImageNet. (b) The image marked with relevant keypoints. (c) The weighting of keypoints by their response strength, where the weights are sorted in order to show the range and distribution of probabilities. (d) An illustration of the probability distribution after observing one million samples.

Over the years, many artificial visual systems have been proposed with varying degrees of success. Modern convolutional neural networks (CNNs) are hypothesized to model the primary visual cortex of humans and primates. The success of modern networks in addition to observations of their hidden layers has been cited as support for this hypothesis. Prior to the success of CNNs, feature detection was completed by using methods such as the Scale Invariant Feature Transform (SIFT) algorithm.

The methods by which SIFT computes features is not only deterministic, but well understood as a reliable way to identify basic structural and compositional elements of

an image – though it may be difficult or expensive to transform this identification to an understanding of an image's classification. CNN's, on the other hand, are able to successfully understand the mapping between an image and it's classification – but how much of that relationship is dependent on basic structural and compositional elements is unknown. Adversarial examples show that minor changes to such elements can be catastrophic.

Given the opposing strengths of SIFT and CNNs, it seemed natural to explore the relationship between deterministic feature detection of SIFT and automatic, stochastic feature detection of CNNs.

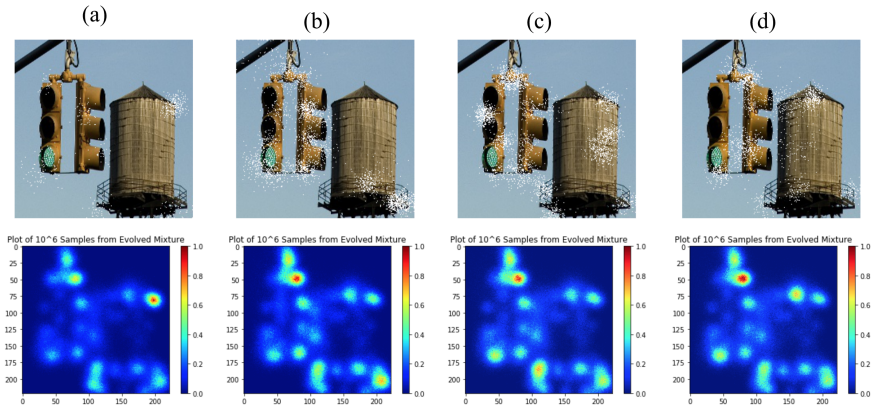


Fig. 15. Illustration of the evolution of a saliency distribution with different target classes. Each of the above images was the result of a single, target Monte Carlo simulation with no refinement procedure for illustrative purposes. (a) Modified to Crane with 57.6% confidence after 572 pixel changes. (b) Modified to Fountain with 48.2% confidence after 2172 pixel changes. (c) Modified to Castle with 20.4% confidence after 2553 pixel changes. (d) Modified to Bell with 27.3% confidence after 1895 pixel changes.

In figure 14 we illustrate an image's transformation into a saliency distribution. While it is intuitive that these two intimately related algorithms have some common ground, it would not be prudent to assume that the feature detection performed by SIFT and by all CNNs is identical. As such, we introduced Monte Carlo saliency updating and MCTS algorithms to actively re-weight the distribution components based on what we are able to glean from querying the CNN model. Constantly updating the saliency distribution – Fig. 14.d – based on the impact of the feature's manipulation wrt the CNN model allows us to dynamically bridge the gap between deterministic and stochastic feature detection (SIFT and CNNs respectively). Further, we can see that, when we allow for the salience distribution to evolve towards a particular target classification, the network weights different keypoints of the image.

Though in figure 15 the differences in saliency distributions are subtle, they are also crucial. Flexible saliency distributions are needed to find and manipulate the most

crucial elements of an image. For example, a set of ideally placed white pixels can cause the network to believe that a structural element is present where it clearly is not. Similarly, a set of ideally placed pixels can disguise key elements which may lead to a misclassification. In either event, one thing seems certain: the key structural and compositional elements are the heart of any visual system. SIFT provides reliable means to pinpoint these elements, which we exploit in our approach.

Compared to CNNs, SIFT is fast but does not generalise; we do not rely on the latter aspect in our approach.

D Information about The Nexar Challenge Network

The retraining of the Nexar network [7] – whose architecture and hyper-parameters are detailed in table 4 – can be achieved by executing the script found in the Examples directory of the SafeCV package. Below we give the training details of the network tested in this paper.

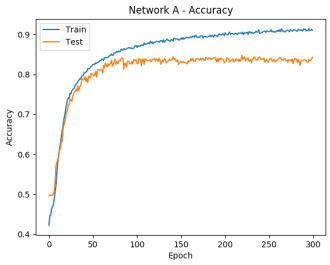


Fig. 16. Accuracy per epoch during the training of the winning entry of the Nexar Challenge [7]; training and test accuracy peek at around 90% and 87% respectively.

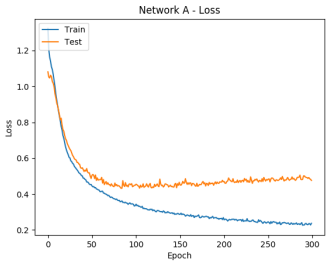


Fig. 17. Loss per epoch during the training of the winning entry of the Nexar Challenge [7].

E Network Architectures In the Experiments

Layer Type	Layer Size	Parameter	SGD
Conv + ReLU	3x3x32	Learning Rate	0.1
Conv + ReLU	3x3x32	Momentum	0.9
Max Pooling	2x2	Delay Rate	-
Conv + ReLU	3x3x64	Dropout	0.5
Conv + ReLU	3x3x64	Batch Size	128
Max Pooling	2x2	Epochs	50
Dense + ReLU	200		
Dense + ReLU	200		
Softmax + ReLU	10		

Table 2. MNIST LeNet Architecture and training parameters used in [8] and [31].

Layer Type	Layer Size	Parameter	Adadelta
Conv + ReLU	3x3x32	Learning Rate	1.0
Conv + ReLU	3x3x64	ρ	0.0
Max Pooling	2x2	Fuzz factor	1e-08
Dense + ReLU	128	Decay	0.0
Softmax + ReLU	10		

Table 3. Architecture and training parameters used by DLV [15]. Implementation provided by [10].

Layer Type	Layer Size	Parameter	Adam
Conv + ReLU	3x3x16	Learning Rate	3e-4
Max Pooling	3x3	Beta 1	0.9
Conv + ReLU	3x3x32	Beta 2	0.999
Max Pooling	3x3	Fuzz Factor	1e-08
Conv + ReLU	3x3x64	Decay	0.0
Max Pooling	2x2		
Dense	128		
Softmax	3		

Table 4. Architecture and training parameters for a winning entry in the Nexar Traffic Light challenge [29].