A Appendix: Proofs of Theorems

A.1 Proof of NP-hardness

Theorem 5 *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a neural network and its input is normalized into $[0,1]^n$. When $||\cdot||_D$ is the L_0 norm, $d_m(f, x_0, ||\cdot||_D)$ is NP-hard, and there at least exists a deterministic algorithm that can compute $d_m(f, x_0, ||\cdot||_D)$ in time complexity $O((1+1/\epsilon)^n)$ for the worst case scenario when the error tolerance for each dimension is $\epsilon > 0$.*

Proof 2 *Here we consider the worst case scenario and use a straight-forward grid search to verify the time complexity needed. In the worst case, the maximum radius of a safe L_0-norm ball for DNN f is $d_m(f, x_0, ||\cdot||_D) = n$. A grid search with grid size $\Delta = 1/\epsilon$ starts from $d_{L_0} = 1$ to verify whether d_{L_0} is the radius of maximum safe L_0-norm ball and would require the following running time in terms of evaluation numbers of the DNN.*

$$\sum_{d_{L_0}=1}^{n} \binom{n}{d_{L_0}} \Delta^{d_{L_0}} = (1+1/\epsilon)^n \tag{20}$$

From the above proof, we get the following remark.

Remark 1 *Computing $d_m(f, x_0, ||\cdot||_D)$ is* more challenging *problem for $D = 0$, since it requires a* higher *computing complexity than $D = 1$ and $D = 2$. Namely, grid search only requires $(1/\epsilon)^n$ evaluations on DNN to estimate $d_m(f, x_0, ||\cdot||_1)$ or $d_m(f, x_0, ||\cdot||_2)$ given the same error tolerance ϵ.*

A.2 Proof of Theorem: Guarantee of Lower Bounds

Proof 3 *Our proof proceeds by contradiction. Let $l = l(f, x_0)$. Assume that there is another adversarial example x_0' such that $t' = ||x_0' - x_0||_0 \leq l$ where t' represents the number of perturbed pixels. By the definition of adversarial examples, there exists a subspace $X_k \in \mathbb{R}^{t'}$ such that $cl(f, \mathcal{S}(x_0, t')[:, k]) \neq cl(f, x_0)$. By $t' \leq l$, we can find a subspace $Y_q \in \mathbb{R}^l$ such that $X_k \subset Y_q$. Thus we have $S(Y_q, l) \geq S(X_k, t')$. Moreover, by $S(Y_q, l) \leq S(Y_1, l)$, we have $cl(f, \mathcal{S}(x_0, l)[:, 1]) \neq cl(f, x_0)$ since $cl(f, \mathcal{S}(x_0, l)[:, p]) \neq cl(f, x_0)$. However, this conflicts with $cl(f, \mathcal{S}(x_0, l)[:, 1]) = cl(f, x_0)$, which can be obtained by the algorithm for computing lower bounds in Section 4.2.*

A.3 Proof of Theorem: Guarantee of Upper Bounds

Proof 4 (Monotonic Decrease Property of Upper Bounds) *We use mathematical induction to prove that upper bounds monotonically decrease.*

Base Case *$m = 1$: Based on the algorithm in* Upper Bounds *of Section 4.2, we assume that, after $m = 1$ subspace perturbations, we find the adversarial example x' such that $cl(f, x') \neq cl(f, x_0)$.*

We know that, at $t = i$, based on the algorithm, we get $S(x_0, i)$ and $\mathcal{S}(x_0, i)$, the ordered subspace sensitivities and their corresponding subspaces. Assume that the ordered subspace list is $\{\phi_1, \phi_2, ..., \phi_m\}$. Then, from the assumption, we have $cl(f, x(\phi_1)) \neq$

$cl(f, x_0)$ *where* $x(\phi_1)$ *denotes the input of the neural network corresponding to subspace* ϕ_1.

Then, at $t = i+1$*, based on the algorithm, we calculate* $S(x_0, i+1)$ *and* $\mathcal{S}(x_0, i+1)$*. Similarly, we assume the ordered subspace list is* $\{\theta_1, \theta_2, ..., \theta_n\}$*. Thus we can find a subspace* θ_q *in* $\{\theta_1, \theta_2, ..., \theta_n\}$ *such that* $\phi_1 \subset \theta_q$*. As a result, we know that* $S(\phi_1) \leq S(\theta_q)$*, thus* $cl(f, x(\theta_q)) \neq cl(f, x_0)$*. After exhaustive tightening process, we can at least find its subset* ϕ_1*, since* $cl(f, \phi_1) \neq cl(f, x_0)$ *still holds after removing the pixels* $x = \theta_q - \phi_1$*. So we know* $||x(\theta_q) - x_0||_0 \leq ||x(\phi_1) - x_0||_0$*. However,* θ_q *will not necessarily be found at* $t = i + 1$ *in our upper bound algorithm, depending on its location in* $\{\theta_1, \theta_2, ..., \theta_n\}$*:*

1. If it is in the front position such as $\{\theta_q, \theta_2, ..., \theta_m\}$*, then we know that* $||x(\theta_q) - x_0||_0 \leq ||x(\phi_1) - x_0||_0$ *based on the above analysis, i.e.,* $u_i(f, x_0) \geq u_{i+1}(f, x_0)$ *holds.*

2. If it is in the behind position such as $\{\theta_1, \theta_2, \theta_q, ..., \theta_m\}$*, we know that* $S(\theta_1) \geq S(\theta_q) \geq S(\phi_1)$*, which means that subspace* θ_1 *leads to a larger network's confidence decrease than subspace* θ_q*, thus* $||x(\theta_1) - x_0||_0 \leq ||x(\theta_q) - x_0||_0$*. We already know* $||x(\theta_q) - x_0||_0 \leq ||x(\phi_1) - x_0||_0$ *thus* $u_i(f, x_0) \geq u_{i+1}(f, x_0)$ *also holds.*

Inductive Case $m = k$*: Assume that at* $t = i$*, after going through* $m = k$ *subspace perturbations, i.e.,* $\Phi_k = \{\phi_1 \cup \phi_2 \cup ... \cup \phi_k\}$*, we find an adversarial example* x' *and* $u_i(f, x_0) \geq u_{i+1}(f, x_0)$ *holds. We need to show that after going through* $m = k + 1$ *subspace perturbations, i.e.,* $\Phi_{k+1} = \{\phi_1 \cup \phi_2 \cup ... \cup \phi_k \cup \phi_{k+1}\}$*, the relation* $u_i(f, x_0) \geq u_{i+1}(f, x_0)$ *also holds.*

Similarly, at $t = i + 1$*, we can find a* k *subspace set* $\Theta_k = \{\theta_{q_1} \cup \theta_{q_2} \cup ... \cup \theta_{q_k}\}$ *such that* $\Phi_k \subset \Theta_k$*, and we know that* $u_{i+1}(f, x_0) = ||x(\Phi_k) - x_0||_0 \leq ||x(\Theta_k) - x_0||_0 = u_i(f, x_0)$ *holds. Then, for the new subspace* ϕ_{k+1} *at* $t = i$*, we can also find* $\theta_{q_{k+1}}$ *at* $t = i + 1$ *such that* $\phi_{k+1} \subset \theta_{q_{k+1}}$*. We get* $\Theta_{k+1} = \Theta_k \cup \theta_{q_{k+1}}$*. As a result, we still have* $\Phi_{k+1} \subset \Theta_{k+1}$*, obviously,* $cl(x(\Theta_{k+1}), f) \neq cl(x(\Phi_{k+1}), f)$*, which means we can definitely find an adversarial example after all perturbations in* Θ_{k+1}*. After exhaustive tightening process, we can at least find its subset* Φ_{k+1}*, since* $cl(f, x(\Phi_{k+1})) \neq cl(f, x_0)$ *still holds after removing those pixels* $x = \Theta_k - \phi_k$*. So we know* $||x(\Theta_{k+1}) - x_0||_0 \leq ||x(\Phi_{k+1}) - x_0||_0$*, i.e.,* $u_i(f, x_0) \geq u_{i+1}(f, x_0)$ *holds.*

A.4 Proof of Theorem: Uncertainty Radius Convergence to Zero

Proof 5 (Uncertainty Radius Convergence to Zero: $\lim_{t \to n} U_r(l_i, u_i) = 0$**)** *Based on the definition of* L_0*-norm distance (i.e.,* $0 \leq l_i \leq d_m \leq u_i \leq n$*), we know that* $t \to n \implies l_n \to n \implies U_r(l_i, u_i) = 1/2(u_i - l_i) = n - n = 0$.

B Appendix: The L_0 Norm

We justify on the basis of several aspects that the L_0 norm is worthy of being considered.

Technical Reason The reason why norms other than the L_0 norm are widely used is *mainly technical*: the existing adversarial example generation algorithms [1, 17] proceed by first computing the gradient $\nabla_x J(\theta, x, f(x))$, where $J(\theta, x, f(x))$ is a loss or cost function and θ are the learnable parameters of the network f, and then adapting (in different ways for different algorithms) the input x into x' along the gradient descent direction. To enable this computation, the change to the input x needs to be *continuous and differentiable*. It is not hard to see that, while the L_1, L_2 and L_∞ norms are continuous and differentiable, the L_0 norm is not. Nevertheless, the L_0 norm is an effective and efficient method to quantify a range of adversarial perturbations and should be studied.

Tolerance of Human Perception to L_0 Norm Recently, [17] demonstrates through in-situ human experiments that the L_0 norm is good at approximating human visual perception. Specifically, 349 human participants were recruited for a study of visual perception on L_0 image distortions, with the result concluding that nearly all participants can correctly recognise L_0 perturbed images when the rate of distortion pixels is less than 5.61% (i.e., 44 pixels for MNIST and 57 pixels for CIFAR-10) and 90% of them can still recognise them when the distortion rate is less than 14.29% (i.e., 112 pixels for MNIST and 146 pixels for CIFAR-10). This experiment essentially demonstrates that human perception is tolerant of perturbations with only a few pixels changed, and shows the necessity of robustness evaluation based on the L_0 norm.

Usefulness of Approaches without Gradient From the security point of view, an attacker to a network may not be able to access its architecture and parameters, to say nothing of the gradient $\nabla_x J(\theta, x, f(x))$. Therefore, to evaluate the robustness of a network, we need to consider *black-box* attackers, which can only query the network for classification. For a black-box attacker, an attack (or a perturbation) based on the L_0 norm is to change several pixels, which is arguably easier to initiate than attacks based on other norms, which often require modifications to nearly all the pixels.

Effectiveness of Pixel-based Perturbations Perturbations by minimising the number of pixels to be changed have been shown to be effective. For example, [25, 30] show that manipulating a single pixel is sufficient for the classification to be changed for several networks trained on the CIFAR10 dataset and the Nexar traffic light challenge. Our approach can beat the state-of-the-art pixel based perturbation algorithms by finding tighter upper bounds to the maximum safety radius. As far as we know, this is the first work on finding lower bounds to the maximum safety radius.

C Appendix: Discussion of Application Scenarios

We now summarise possible application scenarios for the method proposed in this paper.

Safety Verification Safety verification [7] is to determine, for a given network f, an input x_0, a distance metric $||\cdot||_D$, and a number d, whether the norm ball $X(f, x_0, ||\cdot||_D, d)$ is safe. Our approach will compute a sequence of lower bounds $\mathcal{L}(x_0)$ and upper bounds $\mathcal{U}(x_0)$ for the maximum safe radius $d_m(x_0)$. For every round $i > 0$, we can claim one of the following cases:

- the norm ball $X(f, x_0, ||\cdot||_D, d)$ is safe when $d \leq \mathcal{L}(x_0)_i$
- the norm ball $X(f, x_0, ||\cdot||_D, d)$ is unsafe when $d \geq \mathcal{U}(x_0)_i$
- the safety of $X(f, x_0, ||\cdot||_D, d)$ is unknown when $\mathcal{L}(x_0)_i < d < \mathcal{U}(x_0)_i$.

As a byproduct, our method can return at least one adversarial image for the second case.

Competitive L_0 Attack We have shown that the upper bounds in $\mathcal{U}(x_0)$ are monotonically decreasing. As a result, the generation of upper bounds can serve as a competitive L_0 attack method.

Global Robustness Evaluation Our method can have an asymptotic convergence to the true global robustness with provable guarantees. As a result, for two neural networks f_1 and f_2 that are trained for the same task (e.g., MNIST, CIFAR-10 or ImageNet) but with different parameters or architectures (e.g., different layer types, layer numbers or hidden neuron numbers), if $R(f_1, ||\cdot||_0) > R(f_2, ||\cdot||_0)$ then we can claim that network f_1 is more robust than f_2 in terms of its resistance to L_0-norm adversarial attacks.

Test Case Generation Recently software coverage testing techniques have been applied to DNNs and several test criteria have been proposed, see e.g., [19, 29]. Each criterion defines a set of requirements that have to be tested for a DNN. Given a test suite, the coverage level of the set of requirements indicates the adequacy level for testing the DNN. The technique in this paper can be conveniently used for coverage-based testing of DNNs.

Real-time Robustness Evaluation By replacing the exhaustive search in the algorithm with random sampling and formulating the subspace as a high-dimensional tensor (to enable parallel computation with GPUs), our method becomes *real-time* (e.g., for a MNIST network, it takes around $0.1s$ to generate an adversarial example). A real-time evaluation can be useful for major safety-critical applications, including self-driving cars, robotic navigation, etc.. Moreover, our method can display in real-time a *saliency map*, visualizing how classification decisions of the network are influenced by pixel-level sensitivities.

D Appendix: Case Study Four: Guiding the Design of Robust DNN Architectures

We trained seven DNNs, named DNN-i for $i \in \{1, \ldots, 7\}$, on the MNIST dataset using the same hardware and software platform and identical training parameters. The DNNs differ in their architecture, i.e., the number of layers and the types of the layers. The architecture matters: while all DNNs achieve 100% accuracy during training, we observe accuracy down to 97.75% on the testing dataset. Details of the models are in Appendix I. We aim to identify architectural choices that affect robustness.

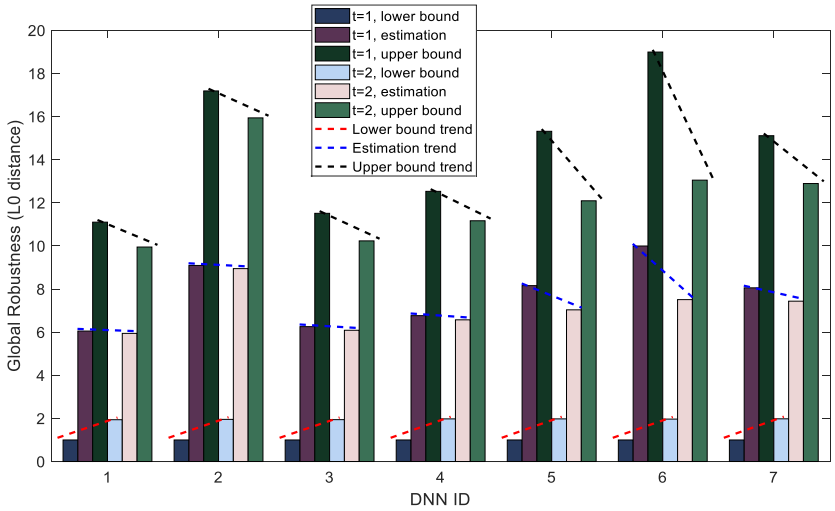


Fig. 8: Lower bounds, upper bounds, and global robustness estimations for $t \in \{1, 2\}$ for seven DNN models

Fig. 8 gives the estimates for global robustness, and their upper and lower bounds at $t = 1$ and $t = 2$ for all seven DNNs. Fig. 9 illustrates the means and standard derivations of the d_m estimates and the uncertainty radius for all 1,000 sampled testing images. And we also find that, the local robustness (i.e., robustness evaluated on a single image) of a network is coincident with its global robustness, so is the uncertainty radius.

We observe the following: *i)* number of layers: a very deep DNN (i.e., too many layers relative to the size of the training dataset) is less robust, such as DNN-7; *ii)* convolutional layers: DNNs with an excessive number of convolutional layers are less robust, e.g., compared with DNN-5, DNN-6 has an additional convolutional layer, but is significantly less robust; *iii)* batch-normalisation layers: adding a batch-normalisation layer may improve robustness, e.g., DNN-3 is more robust than DNN-2.

We remark that testing accuracy is not a good proxy for robustness: a DNN with higher testing accuracy is not necessarily more robust, e.g., DNN-1 and DNN-3 are more robust than DNN-6 and DNN-7, which have higher testing accuracies. DNNs may require

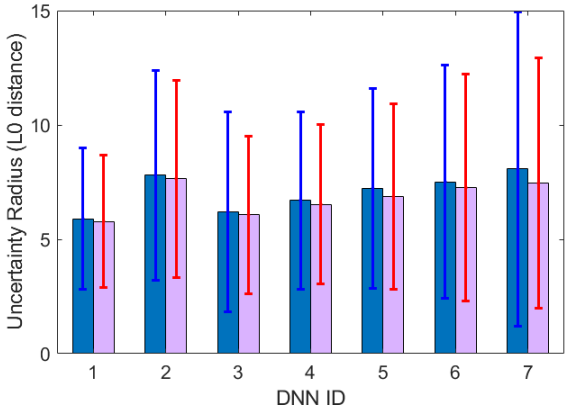


Fig. 9: The means and standard derivations of d_m uncertainty radiuses for 1,000 tested images at $t = 1, 2$.

a balance between robustness and their ability to generalise (proxied by testing accuracy). DNN-4 is a good example, among our limited set.

E Appendix: Case Study Five: Saliency Map and Local Robustness Evaluation for Large-scale ImageNet DNN Models

E.1 Local Robustness Evaluation for ImageNet Models

We apply our method to five state-of-the-art ImageNet DNN models, including AlexNet (8 layers), VGG-16 (16 layers), VGG-19 (19 layers), ResNet50 (50 layers), and ResNet101 (101 layers). We set $t = 1$ and generate the lower/upper bounds and estimates of local robustness for an input image. Fig. 10 gives the local robustness estimates and their bounds for these networks. The adversarial images on the upper boundaries are featured in the top row of Fig. 11. For AlexNet, on this specific image, L0-TRE is able to find its ground-truth adversarial example (local robustness converges at L_0 = 2). We also observe that, for this image, the most robust model is VGG-16 (local robustness = 15) and the most vulnerable one is AlexNet (local robustness = 2). Fig. 10 also reveals that, for similar network structures such as VGG-16 and VGG-19, ResNet50 and ResNet101, a model with deeper layers is less robust to adversarial perturbation. This observation is consistent with our conclusion in Case Study Three.

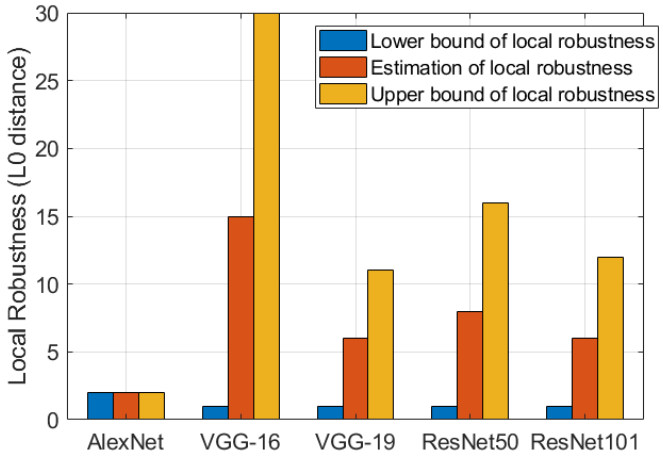


Fig. 10: The upper bound, lower bound and estimation of local robustness for five ImageNet DNNs on a given input image

The method proposed in this paper, just as shown in this case study, provide a possible way to practically evaluate the robustness for large-scale DNN models and such robustness evaluation is with provable bounded guarantees. As a byproduct, our method can also generate saliency map for each input image as shown by the second column of Fig. 11.

E.2 Saliency Map Generation

Model interpretability (or explainability) addresses the problem that the decisions of DNNs are difficult to explain. Recent work, such as [12], calculates the contribution

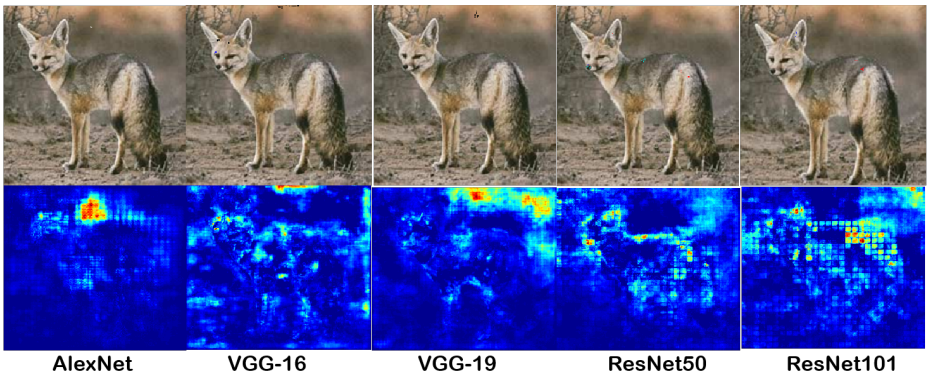


Fig. 11: Adversarial examples on upper boundaries (top) and saliency maps (bottom)

of each input dimension to the output decision. Our computation of subspace sensitivity $S(\mathtt{T}, t)$ can be re-used to quantify this contribution for each pixel of an image.

As shown in Fig. 11 (more examples in Appendix **??**), a brighter area indicates vulnerability to perturbation; it is very easy to see that VGG-16 is the most robust model. We obtain a large bright area for AlexNet, where a minor perturbation can lead to a misclassification. On the contrary, for this image, there are no vulnerable areas for VGG-16. The constraints of the optimisation problem given in Definition 3 can be adapted to generate full saliency maps for the hidden neurons, which have potential as an explanation for the decisions that DNNs make [16]. Moreover, a classic concept in cooperative game theory is to calculate the contribution of the players to their cooperation. Quantifying such contribution values play an important role in game-based safety verification on DNNs such as recent works in [30, 31]. The intermediate result from L0-TRE tool in terms of subspace sensitivity $S(\mathtt{T}, t)$, as shown in this case study, is well suitable for this purpose as validated in paper [31].

F Appendix: Experimental Settings for Case Study One

F.1 Model Structures of sDNN

Table 2: sDNN

Layer Type	Size
Input layer	$14 \times 14 \times 1$
Convolution layer	$2 \times 2 \times 8$
Batch-Normalization layer	8 channels
ReLU activation	
Convolution layer	$2 \times 2 \times 16$
Batch-Normalization layer	16 channels
ReLU activation	
Convolution layer	$2 \times 2 \times 32$
Batch-Normalization layer	32 channels
ReLU activation	
Fully Connected	10
softmax + Class output	

F.2 Parameter Settings of sDNN

Model Training Setup

- Hardware: Notebook PC with I7-7700HQ, 16GB RAM, GTX 1050 GPU
- Software: Matlab 2018a, Neural Network Toolbox, Image Processing Toolbox, Parallel Computing Toolbox
- Parameter Optimization Settings: SGDM, Max Epochs = 20, Mini-Batch Size = 128
- Training Dataset: MNIST training dataset with 50,000 images
- Training Accuracy: 99.5%
- Testing Dataset: MNIST testing dataset with 10,000 images
- Testing Accuracy: 98.73%

Algorithm Setup

- $\epsilon = 0.25$
- Maximum $t = 3$
- Tested Images: 5,300 images sampled from MNIST testing dataset

F.3 Model Structures of DNN-0

F.4 Parameter Settings of DNN-0

Model Training Setup

Table 3: DNN-0

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
ReLU activation	
Maxpooling layer	2×2
Dropout layer	0.25
Fully Connected layer	128
ReLU activation	
Dropout layer	0.5
Fully Connected layer	10
Softmax + Class output	

- Hardware: Notebook PC with I7-7700HQ, 16GB RAM, GTX 1050 GPU
- Software: Matlab 2018a, Neural Network Toolbox, Image Processing Toolbox, Parallel Computing Toolbox
- Parameter Optimization Settings: SGDM, Max Epochs = 30, Mini-Batch Size = 128
- Training Dataset: MNIST training dataset with 50,000 images
- Training Accuracy: 100%
- Testing Dataset: MNIST testing dataset with 10,000 images
- Testing Accuracy: 99.16%

Algorithm Setup

- $\epsilon = 0.25$
- Maximum $t = 2$
- Tested Images: 2,400 images sampled from MNIST testing dataset

F.5 Ground-Truth Adversarial Images

Fig. 12 displays some adversarial images returned by our upper bound algorithm. For each digital image, from the left to right, the first is original image, the second is the adversarial image returned at $t = 1$, and the third is the adversarial example at the boundary of a safe norm ball.

Fig. 12: Ground truth adversarial examples when converging to d_m.

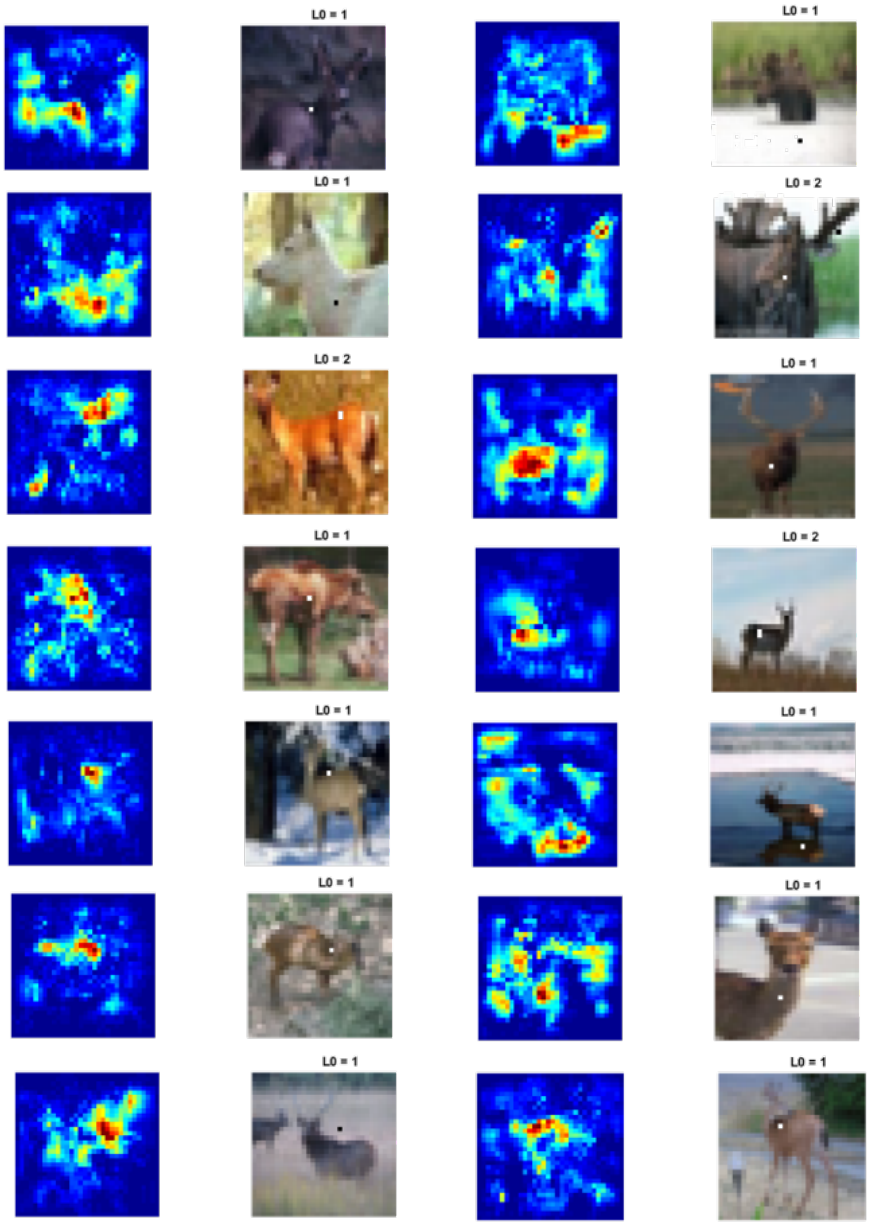


Fig. 13: Ground-truth adversarial examples (right column) generated by L0-TRE at $t = 1$, and the saliency maps of the original images (left column).

G Appendix: Experimental Settings for Case Study Two

G.1 Model Structures for L_0 Attack

The architectures for the MNIST and CIFAR-10 models used in L_0 attack are illustrated in Table 4.

Table 4: Model architectures for the MNIST and CIFAR-10 models.

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

G.2 Model Training Setups

- Parameter Optimization Option: Batch Size = 128, Epochs = 50, Loss Function = tf.nn.softmax_cross_entropy_with_logits, Optimizer = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
- Training Accuracy:
  - MNIST (99.99% on 60,000 images)
  - CIFAR-10 (99.83% on 50,000 images)
- Testing Accuracy:
  - MNIST (99.36% on 10,000 images)
  - CIFAR-10 (78.30% on 10,000 images)

G.3 Adversarial Images

Fig. 14 and 15 present a few adversarial examples generated on the MNIST and CIFAR-10 datasets by our approach L0-TRE, together with results for four other tools, i.e., C&W, JSMA, DLV, and SafeCV.

G.4 Experimental Setting for Competitive L0 Attack Comparison

Baseline Methods We choose four well-established baseline methods that can perform state-of-the-art L_0 adversarial attacks. Their code is available on Github.

– JSMA[7]: is a targeted attack based on the L_0-norm, here used so that adversarial examples are misclassified into all classes except the correct one.
– C&W[8]: is a state-of-the-art adversarial attack method, which models the attack problem as an unconstrained optimization problem that is solvable by the Adam optimizer in Tensorflow.
– DLV[9]: is an untargeted DNN verification method based on exhaustive search and Monte Carlo tree search (MCTS).
– SafeCV[10]: is a feature-guided black-box safety verification and attack method based on the Scale Invariant Feature Transform (SIFT) for feature extraction, game theory, and MCTS.
– DeepGame[11]: is a two-player turn-based game framework for the verification of deep neural networks with provable guarantees on L_1, L_2 and L_∞-norm distances, but with a slight modification it can be used to perform L_0-norm adversarial attack.

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

– MNIST dataset[12]: is an image dataset of handwritten digits, which contains a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image.
– CIFAR-10 dataset[13]: is an image dataset of 10 mutually exclusive classes, i.e., 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'. It consists of 60,000 32x32 colour images - 50,000 for training, and 10,000 for testing.

Platforms

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

[7] https://github.com/tensorflow/cleverhans/blob/master/cleverhans_tutorials/mnist_tutorial_jsma.py
[8] https://github.com/carlini/nn_robust_attacks
[9] https://github.com/VeriDeep/DLV
[10] https://github.com/matthewwicker/SafeCV
[11] https://github.com/TrustAI/DeepGame
[12] http://yann.lecun.com/exdb/mnist/
[13] https://www.cs.toronto.edu/ kriz/cifar.html

G.5 Algorithm Settings

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

- JSMA:
  - bounds = (0, 1)
  - predicts = 'logits'
- C&W:
  - targeted = False
  - learning_rate = 0.1
  - max_iteration = 100
- DLV:
  - mcts_mode = "sift_twoPlayer"
  - startLayer, maxLayer = -1
  - numOfFeatures = 150
  - featureDims = 1
  - MCTS_level_maximal_time = 30
  - MCTS_all_maximal_time = 120
  - MCTS_multi_samples = 5 (MNIST), 3 (CIFAR-10)
- SafeCV:
  - MANIP = max_manip (MNIST), white_manipulation (CIFAR-10)
  - VISIT_CONSTANT = 1
  - backtracking_constant = 1
  - simulation_cutoff = 100
- DeepGame
  - gameType = 'cooperative'
  - bound = 'ub'
  - algorithm = 'A*'
  - eta = ('L0', 30)
  - tau = 1
- Ours:
  - EPSILON = 0.5
  - L0_UPPER_BOUND = 100

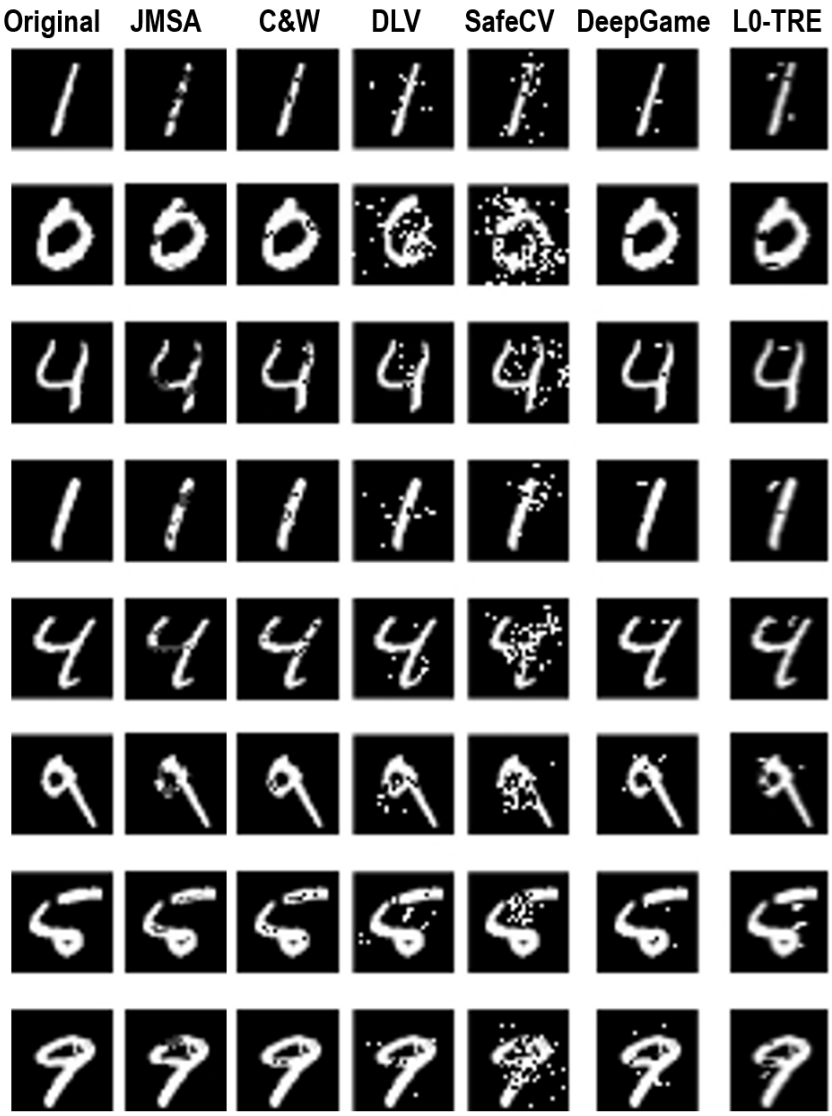


Fig. 14: Adversarial images generated by the L_0 attack methods on the MNIST dataset. From left to right: original image, JSMA, C&W, DLV, SafeCV, DeepGame, and our tool L0-TRE.

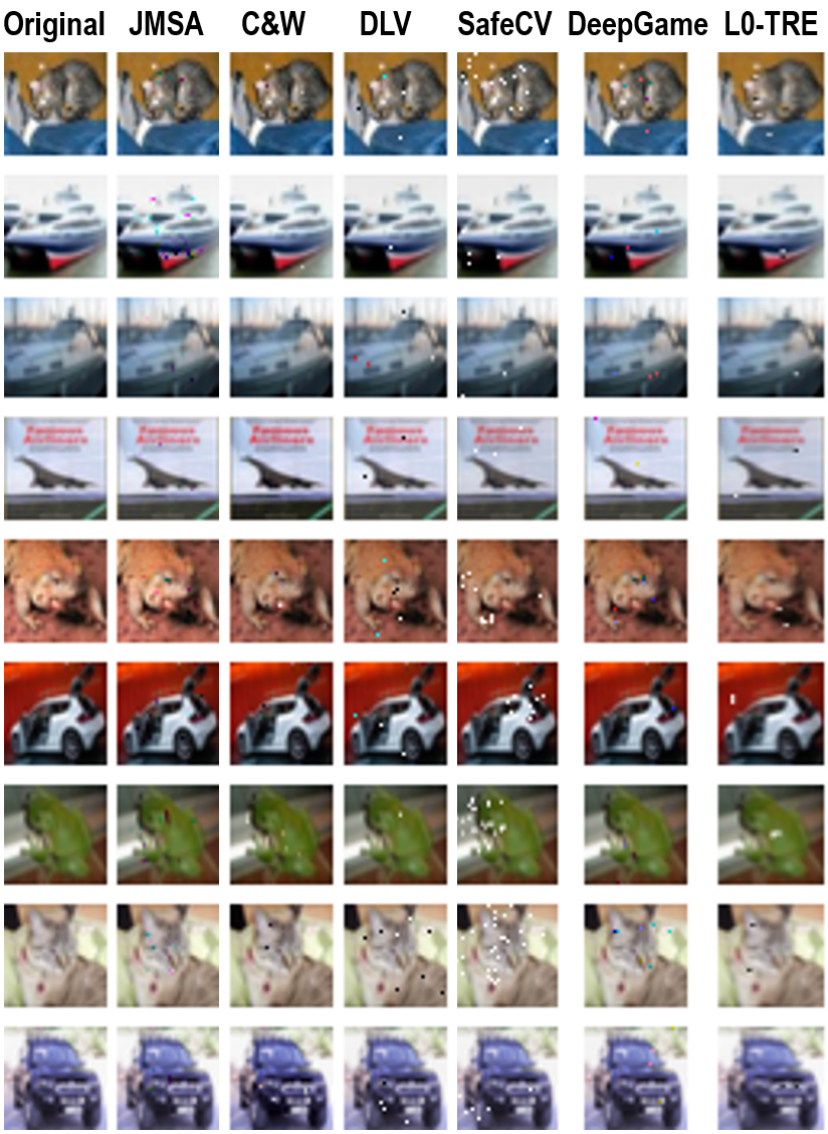
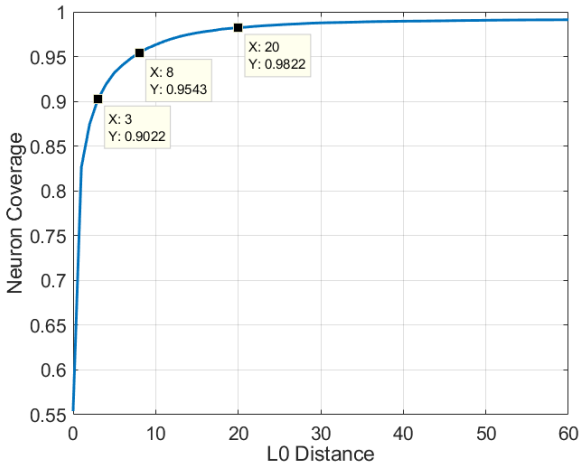


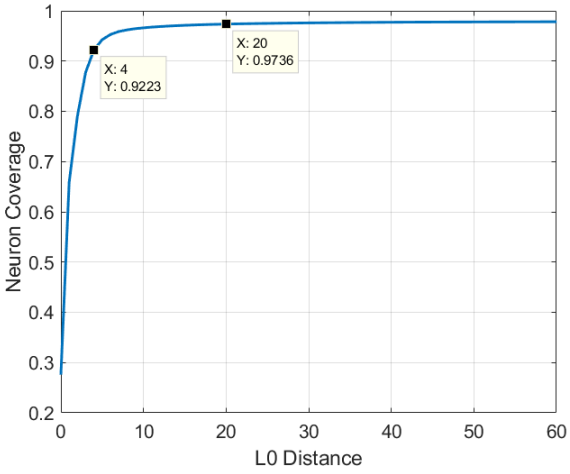
Fig. 15: Adversarial images generated by the L_0 attack methods on the CIFAR-10 dataset. From left to right:original image, JSMA, C&W, DLV, SafeCV, DeepGame, and our tool L0-TRE.

H Appendix: Experimental Settings for Case Study Three

The experimental settings of this case study can be found in Appendix G.



(a) MNIST



(b) CIFAR-10

Fig. 16: Neuron coverage by robustness evaluation on MNIST (a) and CIFAR-10 (b). *The horizontal axis measures the L_0 distance of each generated test case with respect to the original input, and the vertical axis records the coverage percentage. We see that it achieves more than %90 neuron coverage by only modifying 3 pixels.*

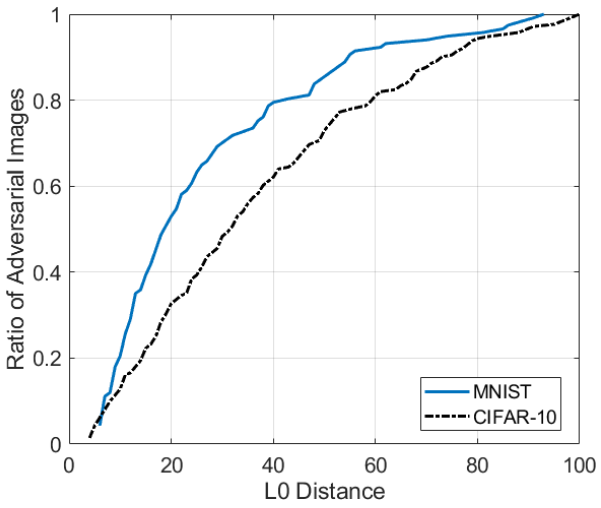


Fig. 17: Neuron coverage: the percentage of adversarial examples within each distance. *It depicts that a significant portion of adversarial examples can be found in the relatively small L_0 distance end of the curve.* More experimental results and applications of applying L0-TRE into DNN testing can be found in GitHub `https://github.com/TrustAI/DeepConcolic`

I Appendix: Experimental Settings for Case Study Four

I.1 Model Training and Algorithm Setup

- Hardware: Notebook PC with I7-7700HQ, 16GB RAM, GTX 1050 GPU
- Software: Matlab 2018a, Neural Network Toolbox, Image Processing Toolbox, Parallel Computing Toolbox
- Parameter Optimization Settings: SGDM, Max Epochs = 30, Mini-Batch Size = 128
- Training Dataset: MNIST training dataset with 50,000 images
- Training Accuracy: All seven models reach 100%
- Testing Dataset: MNIST testing dataset with 10,000 images
- Testing Accuracy: DNN-1 = 97.75%; DNN-2 = 97.95%; DNN-3 = 98.38%; DNN-4 = 99.06%; DNN-5 = 99.16%; DNN-6 = 99.13%; DNN-7 = 99.41%
- L0-TRE Algorithm Setup: $\epsilon = 0.3$, Maximum $t = 2$, Tested Images: 1,000 images sampled from MNIST testing dataset

I.2 Model Structures of DNN-1 to DNN-7

The model structures of DNN-1 to DNN-7 are described in respective tables.

Table 5: DNN-1

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Fully Connected layer	10
Softmax + Class output	

Table 6: DNN-2

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
ReLU activation	
Fully Connected layer	10
Softmax + Class output	

Table 7: DNN-3

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
Batch-Normalization layer	
ReLU activation	
Fully Connected layer	10
Softmax + Class output	

Table 8: DNN-4

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
Batch-Normalization layer	
ReLU activation	
Fully Connected layer	128
ReLU activation	
Fully Connected layer	10
Softmax + Class output	

Table 9: DNN-5

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
Batch-Normalization layer	
ReLU activation	
Dropout layer	0.5
Fully Connected layer	128
ReLU activation	
Fully Connected layer	10
Softmax + Class output	

Table 10: DNN-6

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 32$
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
ReLU activation	
Convolution layer	$3 \times 3 \times 128$
Batch-Normalization layer	
ReLU activation	
Dropout layer	0.5
Fully Connected layer	128
ReLU activation	
Fully Connected layer	10
Softmax + Class output	

Table 11: DNN-7

Layer Type	Size
Input layer	$28 \times 28 \times 1$
Convolution layer	$3 \times 3 \times 16$
ReLU activation	
Convolution layer	$3 \times 3 \times 32$
Batch-Normalization layer	
ReLU activation	
Convolution layer	$3 \times 3 \times 64$
Batch-Normalization layer	
ReLU activation	
Convolution layer	$3 \times 3 \times 128$
Batch-Normalization layer	
ReLU activation	
Dropout layer	0.5
Fully Connected layer	256
ReLU activation	
Dropout layer	0.5
Fully Connected layer	10
Softmax + Class output	

J Appendix: Experimental Settings for Case Study Five

J.1 State-of-the-art ImageNet DNN Models

- AlexNet [9] : a convolutional neural network, which was originally designed in the ImageNet Large Scale Visual Recognition Challenge in 2012. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up.
- VGG-16 and VGG-19 [24] : were released in 2014 by the Visual Geometry Group at the University of Oxford. This family of architectures achieved second place in the 2014 ImageNet Classification competition, achieving 92.6% top-five accuracy on the ImageNet 2012 competition dataset.
- ResNet50 and ResNet101 [6] : are designed based on residual nets with a depth of 50 and 101 layers. An ensemble of these networks achieved 3.57% testing error on ImageNet and won the 1st place on the ILSVRC 2015 classification task. Their variants also won 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

J.2 Experimental Settings

Platforms

- Hardware: Notebook PC with I7-7700HQ, 16GB RAM, GTX 1050 GPU
- Software: Matlab 2018a, Neural Network Toolbox, Image Processing Toolbox, Parallel Computing Toolbox, and AlexNet, VGG-16, VGG-19, ResNet50 and ResNet101 Pretrained DNN Models

Algorithm Setup

- $\epsilon = 0.3$
- Maximum $t = 1$
- Tested Images: 20 ImageNet images, randomly chosen

J.3 Adversarial Images and Saliency Maps

Fig. 18 gives more examples of adversarial images and saliency maps.

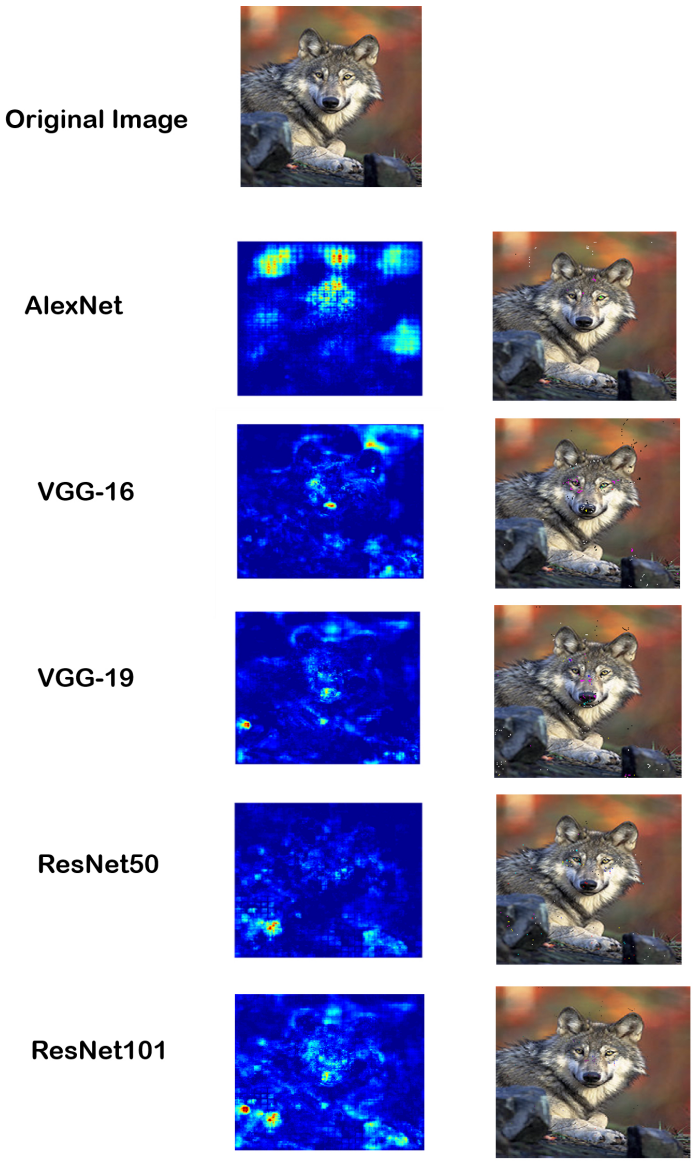


Fig. 18: Adversarial examples on upper boundaries returned by our tool L0-TRE (right column), and saliency maps for each ImageNet DNN model (left column).

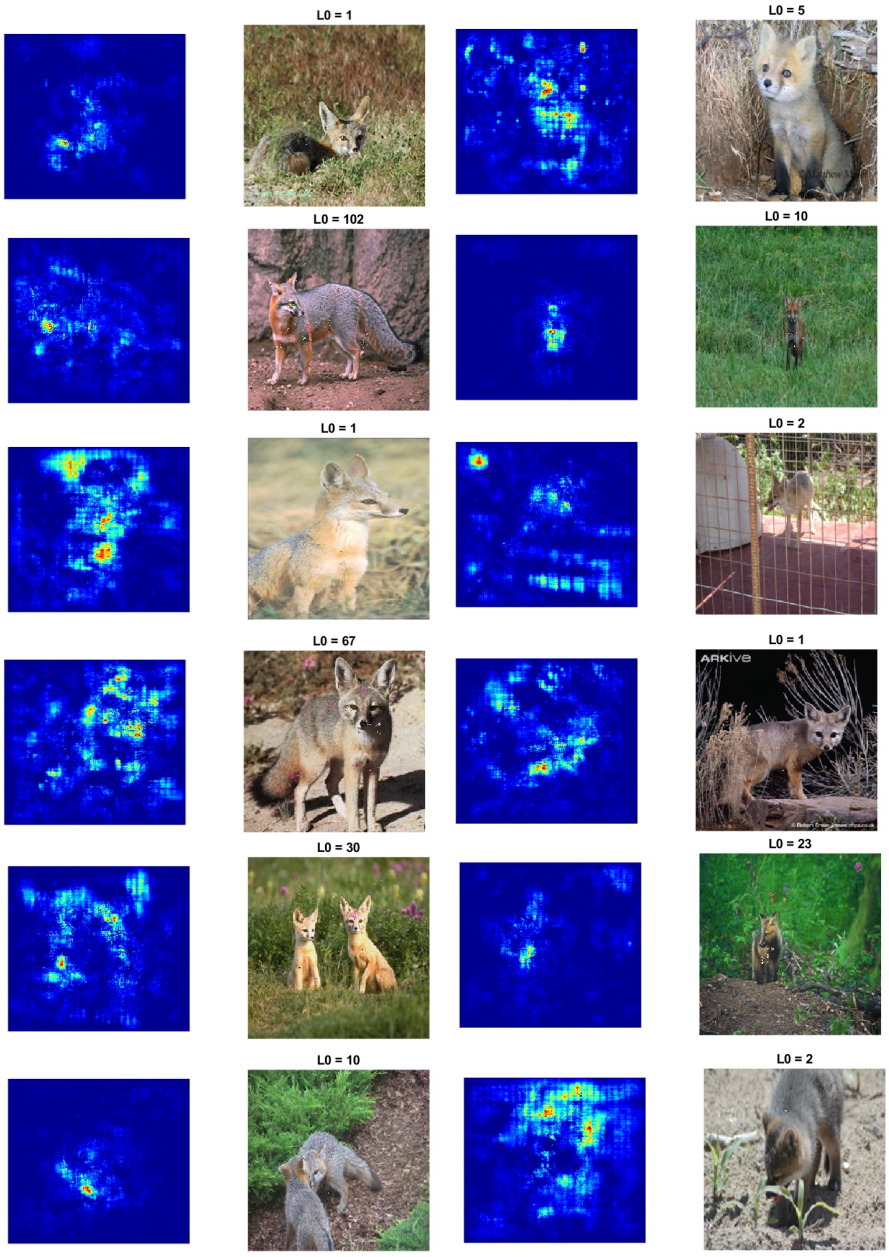


Fig. 19: Adversarial examples on upper boundaries (right column) and their saliency maps (left column) for ImageNet AlexNet DNNs. Note that all adversarial images with $L_0 = 1, 2$ are also the ground-truth L_0-norm adversarial images since their upper bounds and lower bounds local robustness have converged.

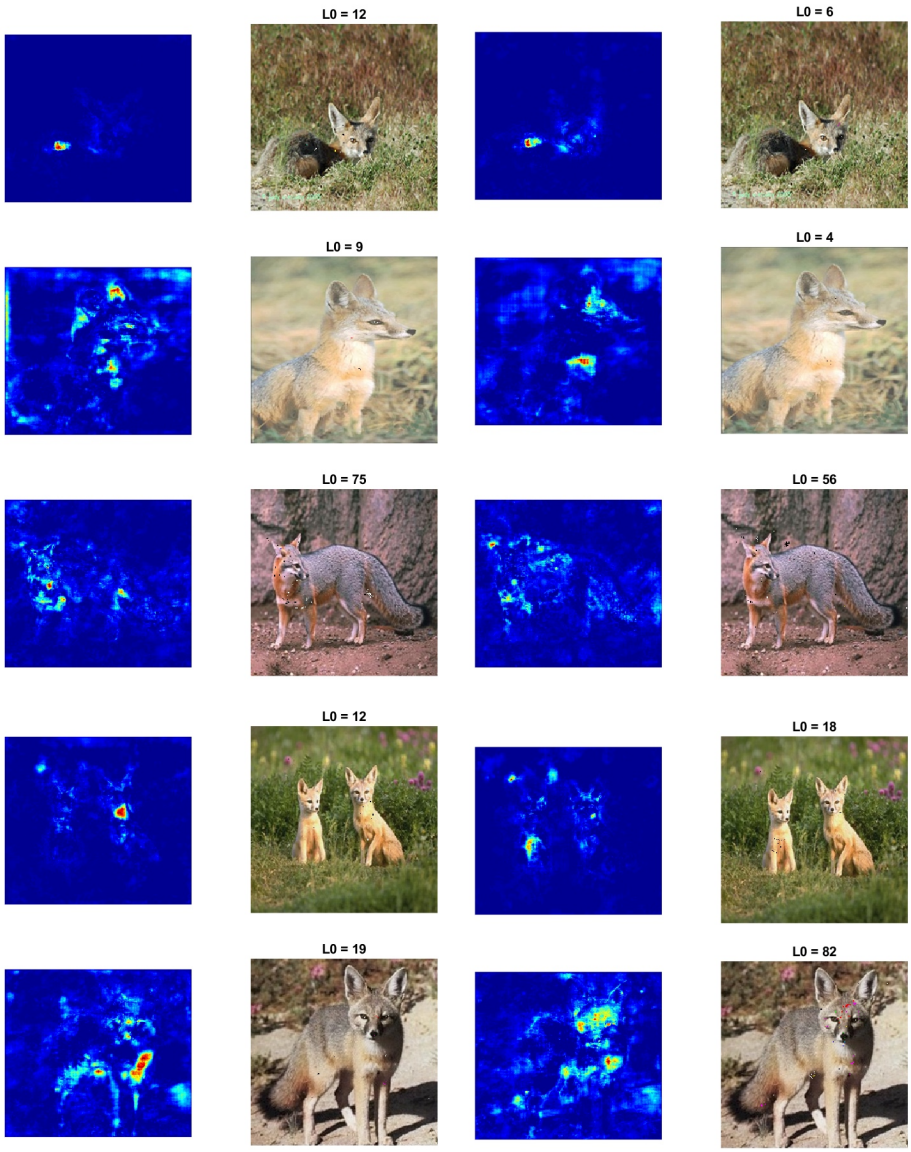


Fig. 20: Adversarial examples on upper boundaries (right column) returned by L0-TRE, and their saliency maps (left column) for VGG-16 and VGG-19. The first and second columns are for VGG-16; the third and fourth columns are for VGG-19.