usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

Interpretable Deep Learning under Fire

Xinyang Zhang, *Pennsylvania State University;* Ningfei Wang, *University of California Irvine;* Hua Shen, *Pennsylvania State University;* Shouling Ji, *Zhejiang University and Alibaba-ZJU Joint Institute of Frontier Technologies;* Xiapu Luo, *Hong Kong Polytechnic University;* Ting Wang, *Pennsylvania State University*

https://www.usenix.org/conference/usenixsecurity20/presentation/zhang-xinyang

This paper is included in the Proceedings of the
29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the
29th USENIX Security Symposium
is sponsored by USENIX.

Interpretable Deep Learning under Fire

Xinyang Zhang[*] Ningfei Wang[⋆] Hua Shen[*] Shouling Ji[†] Xiapu Luo[‡] Ting Wang[*]
[*]*Pennsylvania State University* [⋆]*University of California Irvine*
[†]*Zhejiang University and Alibaba-ZJU Joint Institute of Frontier Technologies*
[‡]*Hong Kong Polytechnic University*

Abstract

Providing explanations for deep neural network (DNN) models is crucial for their use in security-sensitive domains. A plethora of interpretation models have been proposed to help users understand the inner workings of DNNs: how does a DNN arrive at a specific decision for a given input? The improved interpretability is believed to offer a sense of security by involving human in the decision-making process. Yet, due to its data-driven nature, the interpretability itself is potentially susceptible to malicious manipulations, about which little is known thus far.

Here we bridge this gap by conducting the first systematic study on the security of interpretable deep learning systems (IDLSes). We show that existing IDLSes are highly vulnerable to adversarial manipulations. Specifically, we present Adv^2, a new class of attacks that generate adversarial inputs not only misleading target DNNs but also deceiving their coupled interpretation models. Through empirical evaluation against four major types of IDLSes on benchmark datasets and in security-critical applications (e.g., skin cancer diagnosis), we demonstrate that with Adv^2 the adversary is able to arbitrarily designate an input's prediction and interpretation. Further, with both analytical and empirical evidence, we identify the prediction-interpretation gap as one root cause of this vulnerability – a DNN and its interpretation model are often misaligned, resulting in the possibility of exploiting both models simultaneously. Finally, we explore potential countermeasures against Adv^2, including leveraging its low transferability and incorporating it in an adversarial training framework. Our findings shed light on designing and operating IDLSes in a more secure and informative fashion, leading to several promising research directions.

1 Introduction

The recent advances in deep learning have led to breakthroughs in many long-standing machine learning tasks (e.g., image classification [22], natural language processing [54], and even playing Go [49]), enabling use cases previously considered strictly experimental.

However, the state-of-the-art performance of deep neural network (DNN) models is often achieved at the cost of interpretability. It is challenging to intuitively understand the inference of complicated DNNs – how does a DNN arrive at a specific decision for a given input – due to their high non-linearity and nested architectures. This is a major drawback for applications in which the interpretability of decisions is a critical prerequisite, while simple black-box predictions cannot be trusted by default. Another drawback of DNNs is their inherent vulnerability to adversarial inputs – maliciously crafted samples to trigger target DNNs to malfunction [9,28,56] – which leads to unpredictable model behaviors and hinders their use in security-sensitive domains.

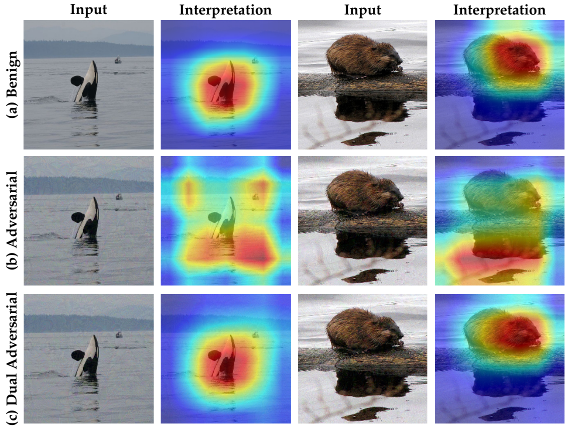


Figure 1: Sample (a) benign, (b) regular adversarial, and (c) dual adversarial inputs and interpretations on ResNet [22] (classifier) and Cam [64] (interpreter).

The drawbacks have spurred intensive research on improving the DNN interpretability via providing explanations at either model-level [26,45,63] or instance-level [10,16,43,50]. For example, in Figure 1 (a), an attribution map highlights an input's most informative part with respect to its classification, revealing their causal relationship. Such interpretability helps users understand the inner workings of DNNs, enabling use cases including model debugging [39], digesting security analysis results [20], and detecting adversarial inputs [13]. For instance, in Figure 1 (b), an adversarial input, which causes the target DNN to deviate from its normal behavior, generates an attribution map highly distinguishable from its benign counterpart, and is thus easily detectable.

As illustrated in Figure 2, a DNN model (classifier), coupled with an interpretation model (interpreter), forms an in-

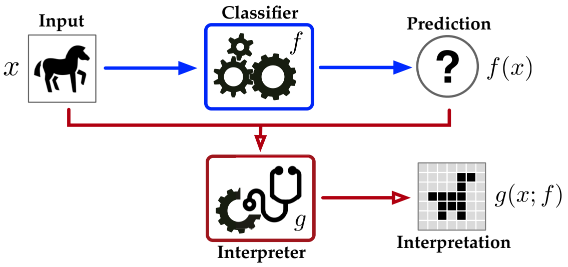


Figure 2: Workflow of an interpretable deep learning system (IDLS).

terpretable deep learning system (IDLS). The enhanced interpretability of IDLSes is believed to offer a sense of security by involving human in the decision process [57]. However, given its data-driven nature, this interpretability itself is potentially susceptible to malicious manipulations. Unfortunately, thus far, little is known about the security vulnerability of IDLSes, not to mention mitigating such threats.

Our Work. To bridge the gap, in this paper, we conduct a comprehensive study on the security vulnerability of IDLSes, which leads to the following interesting findings.

First, we demonstrate that existing IDLSes are highly vulnerable to adversarial manipulations. We present ADV^2, a new class of attacks that generate adversarial inputs not only misleading a target DNN but also deceiving its coupled interpreter. By empirically evaluating ADV^2 against four major types of IDLSes on benchmark datasets and in security-critical applications (e.g., skin cancer diagnosis), we show that it is practical to generate adversarial inputs with predictions and interpretations arbitrarily chosen by the adversary. For example, Figure 1 (c) shows adversarial inputs that are misclassified by target DNNs and also interpreted highly similarly to their benign counterparts. Thus the interpretability of IDLSes merely provides limited security assurance.

Then, we show that one possible root cause of this attack vulnerability lies in the prediction-interpretation gap: the interpreter is often misaligned with the classifier, while the interpreter's interpretation only partially explains the classifier's behavior, allowing the adversary to exploit both models simultaneously. This finding entails several intriguing questions: (i) what, in turn, is the possible cause of this gap? (ii) how does this gap vary across different interpreters? (iii) what is its implication for designing more robust interpreters? We explore all these key questions in our study.

Further, we investigate the transferability of ADV^2 across different interpreters. We note that it is often difficult to find adversarial inputs transferable across distinct types of interpreters, as they generate interpretations from complementary perspectives (e.g., back-propagation, intermediate representations, input-prediction correspondence). This finding points to training an ensemble of interpreters as one potential countermeasure against ADV^2.

Finally, we present adversarial interpretation distillation (AID), an adversarial training framework which integrates ADV^2 in training interpreters. We show that AID effectively reduces the prediction-interpretation gap and potentially helps improve the robustness of interpreters against ADV^2.

Notation	Definition
f, g	target classifier, interpreter
$x_\circ, x_*$	benign, adversarial input
c_t, m_t	adversary's target class, interpretation
$x[i]$	i-th dimension of x
ε	perturbation magnitude bound
$\|\cdot\|$	vector norm
$\ell_{\text{int}}, \ell_{\text{prd}}, \ell_{\text{adv}}$	interpretation, prediction, overall loss
α	learning rate

Table 1. Symbols and notations.

To our best knowledge, this work represents the first systematic study on the security vulnerability of existing IDLSes. We believe our findings shed light on designing and operating IDLSes in a more secure and informative manner.

Roadmap. The remainder of the paper proceeds as follows. § 2 introduces fundamental concepts; § 3 presents the ADV^2 attack and details its implementation against four major types of interpreters; § 4 empirically evaluates its effectiveness; § 5 explores the fundamental causes of the attack vulnerability and discusses possible countermeasures; § 6 surveys relevant literature; the paper is concluded in § 7.

2 Preliminaries

We begin with introducing a set of fundamental concepts and assumptions. The symbols and notations used in this paper are summarized in Table 1.

Classifier – In this paper, we primarily focus on predictive tasks (e.g., image classification [12]), in which a DNN f (i.e., classifier) assigns a given input x to one of a set of predefined classes $\mathcal{C}$, $f(x) = c \in \mathcal{C}$.

Interpreter – In general, the DNN interpretability can be obtained in two ways: designing interpretable DNNs [45, 62] or extracting post-hoc interpretations. The latter case does not require modifying model architectures or parameters, thereby leading to higher prediction accuracy. We thus mainly consider post-hoc interpretations in this paper. More specifically, we focus on instance-level interpretability [10,16,26,37,38,45,48,50,63], which explains how a DNN f classifies a given input x and uncovers the causal relationship between x and $f(x)$. We assume such interpretations are given in the form of *attribution maps*. As shown in Figure 2, the interpreter g generates an attribution map $m = g(x; f)$, with its i-th element $m[i]$ quantifying the importance of x's i-th feature $x[i]$ with respect to $f(x)$.

Adversarial Attack – DNNs are inherently vulnerable to adversarial inputs, which are maliciously crafted samples to force DNNs to misbehave [36, 56]. Typically, an adversarial input x_* is generated by modifying a benign input $x_\circ$ via pixel perturbation (e.g., PGD [35]) or spatial transformation (e.g., STADV [60]), with the objective of forcing f to misclassify x_* to a target class c_t, $f(x_*) = c_t \neq f(x_\circ)$. To ensure the attack evasiveness, the modification is often constrained to an allowed set (e.g., a norm ball $\mathcal{B}_\varepsilon(x_\circ) = \{x | \|x - x_\circ\|_\infty \leq \varepsilon\}$). Consider PGD, a universal first-order adversarial attack, as a concrete case. At a high level, PGD implements a sequence of project gradient descent on the loss function:

$$x^{(i+1)} = \Pi_{\mathcal{B}_\varepsilon(x_\circ)}\left(x^{(i)} - \alpha\,\mathrm{sgn}\left(\nabla_x \ell_{\mathrm{prd}}\left(f\left(x^{(i)}\right), c_t\right)\right)\right) \quad (1)$$

where Π is the projection operator, α represents the learning rate, the loss function ℓ_{prd} measures the difference of the model prediction $f(x)$ and the class c_t targeted by the adversary (e.g., cross entropy), and $x^{(0)}$ is initialized as $x_\circ$.

Threat Model – Following the line of work on adversarial attacks [9,19,35,56], we assume in this paper a white-box setting: the adversary has complete access to the classifier f and the interpreter g, including their architectures and parameters. This is a conservative and realistic assumption. Prior work has shown that it is possible to train a surrogate model f' given black-box access to a target DNN f [41]; given that the interpreter is often derived directly from the classifier (details in § 3), the adversary may then train a substitution interpreter g' based on f'. We consider investigating such black-box attacks as our ongoing work.

3 ADV^2 Attack

The interpretability of IDLSes is believed to offer a sense of security by involving human in the decision process [13, 17, 20, 57]; this belief has yet to be rigorously tested. We bridge this gap by presenting ADV^2, a new class of attacks that deceive target DNNs and their interpreters simultaneously. Below we first give an overview of ADV^2 and then detail its instantiations against four major types of interpreters.

3.1 Attack Formulation

The ADV^2 attack deceives both the DNN f and its coupled interpreter g. Specifically, ADV^2 generates an adversarial input x_* by modifying a benign input $x_\circ$ such that

- (i) x_* is misclassified by f to a target class c_t, $f(x_*) = c_t$;
- (ii) x_* triggers g to generate a target attribution map m_t, $g(x_*; f) = m_t$;
- (iii) The difference between x_* and $x_\circ$, $\Delta(x_*, x_\circ)$, is imperceptible;

where the distance function Δ depends on the concrete modification: for pixel perturbation (e.g., [35]), it is instantiated as $\mathcal{L}_\mathrm{p}$ norm, while for spatial transformation (e.g., [60]), it is defined as the overall spatial distortion.

In other words, the goal is to find sufficiently small perturbation to the benign input that leads to the prediction and interpretation desired by the adversary.

At a high level, we formulate ADV^2 using the following optimization framework:

$$\min_x \Delta(x, x_\circ) \quad \text{s.t.} \left\{ \begin{array}{l} f(x) = c_t \\ g(x; f) = m_t \end{array} \right. \quad (2)$$

where the constraints ensure that (i) the adversarial input is misclassified as c_t and (ii) it triggers g to generate the target attribution map m_t.

As the constraints of $f(x) = c_t$ and $g(x; f) = m_t$ are highly non-linear for practical DNNs, we reformulate Eqn (2) in a form more suited for optimization:

$$\begin{aligned} \min_x \quad & \ell_{\mathrm{prd}}(f(x), c_t) + \lambda \ell_{\mathrm{int}}(g(x; f), m_t) \\ \text{s.t.} \quad & \Delta(x, x_\circ) \leq \varepsilon \end{aligned} \quad (3)$$

where the prediction loss ℓ_{prd} is the same as in Eqn (1), the interpretation loss ℓ_{int} measures the difference of adversarial map $g(x; f)$ and target map m_t, and the hyper-parameter λ balances the two factors. Below we use $\ell_{\mathrm{adv}}(x)$ to denote the overall loss function defined in Eqn (3).

We construct the solver of Eqn (3) upon an adversarial attack framework. While it is flexible to choose the concrete framework, below we primarily use PGD [35] as the reference and discuss the construction of ADV^2 upon alternative frameworks (e.g., spatial transformation [60]) in § 4.

Under this setting, we define $\ell_{\mathrm{prd}}(f(x), c_t) = -\log(f_{c_t}(x))$ (i.e., the negative log likelihood of x with respect to the class c_t), $\Delta(x, x_\circ) = \|x - x_\circ\|_\infty$, and $\ell_{\mathrm{int}}(g(x; f), m_t) = \|g(x; f) - m_t\|_2^2$. In general, ADV^2 searches for x_* using a sequence of gradient descent updates:

$$x^{(i+1)} = \Pi_{\mathcal{B}_\varepsilon(x_\circ)}\left(x^{(i)} - \alpha\,\mathrm{sgn}\left(\nabla_x \ell_{\mathrm{adv}}\left(x^{(i)}\right)\right)\right) \quad (4)$$

However, directly applying Eqn (4) is often found ineffective, due to the unique characteristics of individual interpreters. In the following, we detail the instantiations of ADV^2 against the back-propagation-, representation-, model-, and perturbation-guided interpreters, respectively.

3.2 Back-Propagation-Guided Interpretation

This class of interpreters compute the gradient (or its variants) of the model prediction with respect to a given input to derive the importance of each input feature. The hypothesis is that larger gradient magnitude indicates higher relevance of the feature to the prediction. We consider gradient saliency (GRAD) [50] as a representative of this class.

Intuitively, GRAD considers a linear approximation of the model prediction (probability) $f_c(x)$ for a given input x and a given class c, and derives the attribution map m as:

$$m = \left| \frac{\partial f_c(x)}{\partial x} \right| \quad (5)$$

To attack GRAD-based IDLSes, we may search for x_* using a sequence of gradient descent updates as defined in Eqn (4). However, according to Eqn (5), computing the gradient of the attribution map $g(x; f)$ amounts to computing the Hessian matrix of $f_c(x)$, which is all-zero for DNNs with ReLU activation functions. Thus the gradient of the interpretation loss ℓ_{int} provides little information for updating x, which makes directly applying Eqn (4) ineffective.

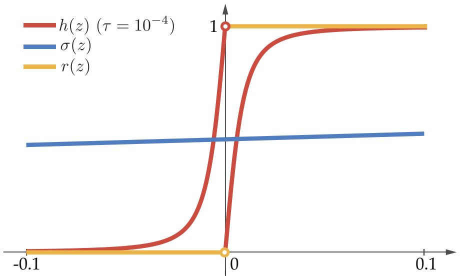


Figure 3: Comparison of $h(z)$, $\sigma(z)$, and $r(z)$ near $z = 0$.

To overcome this, when performing back-propagation, we smooth the gradient of ReLU, denoted by $r(z)$, with a function $h(z)$ defined as (τ is a small constant, e.g., 10^{-4}):

$$h(z) \triangleq \begin{cases} (z+\sqrt{z^2+\tau})' = 1+z/\sqrt{z^2+\tau} & (z<0) \\ (\sqrt{z^2+\tau})' = z/\sqrt{z^2+\tau} & (z \geq 0) \end{cases}$$

Intuitively, $h(z)$ tightly approximates $r(z)$, while its gradient is non-zero everywhere. Another possibility is the sigmoid function $\sigma(z) = 1/(1+e^{-z})$. Figure 3 compares different functions near $z = 0$. Our evaluation shows that $h(z)$ significantly outperforms $\sigma(z)$ and $r(z)$ in attacking GRAD.

This attack is extensible to other back-propagation-based interpreters (e.g., DEEPLIFT [48], SMOOTHGRAD [51], and LRP [6]), due to their fundamentally equivalent, gradient-centric formulations [3].

3.3 Representation-Guided Interpretation

This class of interpreters leverage the feature maps at intermediate layers of DNNs to generate attribution maps. We consider class activation mapping (CAM) [64] as a representative interpreter of this class.

At a high level, CAM performs global average pooling [30] over the feature maps of the last convolutional layer, and uses the outputs as features for a linear layer with softmax activation to approximate the model predictions. Based on this connectivity structure, CAM computes the attribution maps by projecting the weights of the linear layer back to the convolutional feature maps.

Formally, let $a_k[i, j]$ denote the activation of the k-th channel of the last convolutional layer at the spatial position (i, j). The output of global average pooling is defined as $A_k = \sum_{i,j} a_k[i, j]$. Further let $w_{k,c}$ be the weight of the connection between the k-th input and the c-th output of the linear layer. The input to the softmax function for a class c with respect to a given input x is approximated by:

$$z_c(x) \approx \sum_k w_{k,c} A_k = \sum_{i,j} \sum_k w_{k,c} a_k[i, j] \tag{6}$$

The class activation map m_c is then given by:

$$m_c[i, j] = \sum_k w_{k,c} a_k[i, j] \tag{7}$$

Due to its use of deep representations at intermediate layers, CAM generates attribution maps of high visual quality and limited noise and artifacts [30].

We instantiate g with a DNN that concatenates the part of f up to its last convolutional layer and a linear layer parameterized by $\{w_{k,c}\}$. To attack CAM, we search for x_* using a sequence of gradient descent updates as defined in Eqn (4). This attack can be readily extended to other representation-guided interpreters (e.g., GRADCAM [47]).

3.4 Model-Guided Interpretation

Instead of relying on deep representations at intermediate layers, model-guided methods train a meta-model to directly predict the attribution map for any given input in a single feed-forward pass. We consider RTS [10] as a representative method in this category.

For a given input x in a class c, RTS finds its attribution map m by solving the following optimization problem:

$$\begin{aligned} \min_m \quad & \lambda_1 r_{\text{tv}}(m) + \lambda_2 r_{\text{av}}(m) - \log\left(f_c\left(\phi(x; m)\right)\right) \\ & +\lambda_3 f_c\left(\phi(x; 1-m)\right)^{\lambda_4} \\ \text{s.t.} \quad & 0 \leq m \leq 1 \end{aligned} \tag{8}$$

Here $r_{\text{tv}}(m)$ denotes the total variation of m, which reduces noise and artifacts in m; $r_{\text{av}}(m)$ represents the average value of m, which minimizes the size of retained parts; $\phi(x;m)$ is the operator using m as a mask to blend x with random colors and Gaussian blur, which captures the impact of retained parts (where the mask is non-zero) on the model prediction; the hyper-parameters $\{\lambda_i\}_{i=1}^4$ balance these factors. Intuitively, this formulation finds the sufficient and necessary parts of x, based on which f is able to make the prediction $f(x)$ with high confidence.

However, solving Eqn (8) for every input during inference is fairly expensive. Instead, RTS trains a DNN to directly predict the attribution map for any given input, without accessing to the DNN f after training. In [44], this is achieved by composing a ResNet [22] pre-trained on ImageNet [12] as the encoder (which extracts feature maps of given inputs at different scales) and a U-NET [44] as the masking model, which is then trained to directly optimize Eqn (8). We consider the composition of this encoder and this masking model as the interpreter g.

To attack RTS, one may directly apply Eqn (4). However, our evaluation shows that this strategy is often ineffective for finding desirable adversarial inputs. This is explained by that the encoder enc$(\cdot)$ plays a significant role in generating attribution maps, while solely relying on the outputs of the masking model is insufficient to guide the attack. We thus add to Eqn (3) an additional loss term $\ell_{\text{enc}}(\text{enc}(x), \text{enc}(c_t))$, which measures the difference of the encoder's outputs for the adversarial input x and the target class c_t.

We then search for the adversarial input x_* with a sequence of gradient descent updates defined in Eqn (4). More implementation details are discussed in § 3.6.

3.5 Perturbation-Guided Interpretation

The fourth class of interpreters formulate finding the attribution map by perturbing the input with minimum noise and observing the change in the model prediction. We consider MASK [16] as a representative interpreter in this class.

For a given input x, MASK identifies its most informative parts by checking whether changing such parts influences the prediction $f(x)$. It learns a mask m, where $m[i] = 0$ if the i-th input feature is retained and $m[i] = 1$ if the feature is replaced with Gaussian noise. The optimal mask is found by solving an optimization problem:

$$\min_m \; f_c(\phi(x;m)) + \lambda \|1 - m\|_1 \quad \text{s.t. } 0 \le m \le 1 \tag{9}$$

where c denotes the current prediction $c = f(x)$ and $\phi(x;m)$ is the perturbation operator which blends x with Gaussian noise. The first term finds m that causes the probability of c to decrease significantly, while the second term encourages m to be sparse. Intuitively, solving Eqn (9) amounts to finding the most informative and necessary parts of x with respect to its prediction $f(x)$. Note that this formulation may result in significant artifacts in m. A more refined formulation is given in [16].

Unlike other classes of interpreters, to attack MASK, it is infeasible to directly optimize Eqn (3) with iterative gradient descent (Eqn (4)), because the interpreter g itself is formulated as an optimization procedure.

Instead, we reformulate ADV^2 using a bilevel optimization framework. For given $x_\circ$, c_t, m_t, f, and g, we re-define the adversarial loss function as $\ell_{\text{adv}}(x,m) \triangleq \ell_{\text{prd}}(f(x),c_t) + \lambda \ell_{\text{int}}(m,m_t)$ by introducing m as an additional variable. Let $\ell_{\text{map}}(m;x)$ be the objective function defined in Eqn (9). Note that $m_*(x) = \arg\min_m \ell_{\text{map}}(m;x)$ is the attribution map found by MASK for a given input x. We then have the following attack framework:

$$\begin{aligned} \min_x \quad & \ell_{\text{adv}}\left(x, m_*(x)\right) \\ \text{s.t.} \quad & m_*(x) = \arg\min_m \ell_{\text{map}}(m;x) \end{aligned} \tag{10}$$

Still, solving the bilevel optimization in Eqn (10) exactly is challenging, as it requires recomputing $m_*(x)$ by solving the inner optimization problem whenever x is updated. We propose an approximate iterative procedure which optimizes x and m by alternating between gradient descent on ℓ_{adv} and ℓ_{map} respectively.

More specifically, at the i-th iteration, given the current input $x^{(i-1)}$, we compute its attribution map $m^{(i)}$ by updating $m^{(i-1)}$ with gradient descent on $\ell_{\text{map}}\left(m^{(i-1)}; x^{(i-1)}\right)$; we then fix $m^{(i)}$ and obtain $x^{(i)}$ by minimizing ℓ_{adv} after a single step of gradient descent with respect to $m^{(i)}$. Formally, we define the objective function for updating $x^{(i)}$ as:

$$\ell_{\text{adv}}\left(x^{(i-1)}, m^{(i)} - \xi \nabla_m \ell_{\text{map}}\left(m^{(i)}; x^{(i-1)}\right)\right)$$

where ξ is the learning rate for this virtual gradient descent.

The rationale behind this procedure is as follows. While it is difficult to directly minimizing $\ell_{\text{adv}}(x, m_*(x))$ with respect to x, we use a single-step unrolled map as a surrogate of $m_*(x)$. A similar approach is used in [15]. Essentially, this iterative optimization defines a Stackelberg game [46] between the optimizer for x (leader) and the optimizer for m (follower), which requires the leader to anticipate the follower's next move to reach the equilibrium.

Algorithm 1: ADV^2 against MASK.

```
Input: $x_\circ$: benign input; $c_t$: target class; $m_t$: target map; $f$: target
       DNN; $g$: MASK interpreter
Output: $x_*$: adversarial input
initialize $x$ and $m$ as $x_\circ$ and $g(x_\circ; f)$;
while not converged do
    // update m
    update $m$ by gradient descent along $\nabla_m \ell_{\text{map}}(m;x)$;
    // update x with single-step lookahead
    update $x$ by gradient descent along
      $\nabla_x \ell_{\text{adv}}\left(x, m - \xi \nabla_m \ell_{\text{map}}(m;x)\right)$;
return $x$;
```

Algorithm 1 sketches the attack against MASK. More implementation details are given in § 3.6.

3.6 Implementation and Optimization

Next we detail the implementation of ADV^2 and present a suite of optimizations to improve the attack effectiveness against specific interpreters.

Iterative Optimizer – We build the optimizer based upon PGD [35], which iteratively updates the adversarial input using Eqn (4). By default, we use $\mathcal{L}_\infty$ norm to measure the perturbation magnitude. It is possible to adopt alternative frameworks if other perturbation metrics are considered. For instance, instead of modifying pixels directly, one may generate adversarial inputs via spatial transformation [2, 60], in which the perturbation magnitude is often measured by the overall spatial distortion. We detail and evaluate spatial transformation-based ADV^2 in § 4.

Warm Start – It is observed in our evaluation that it is often inefficient to search for adversarial inputs by running the update steps of ADV^2 (Eqn (4)) from scratch. Rather, first running a fixed number (e.g., 400) of update steps of the regular adversarial attack and then resuming the ADV^2 update steps significantly improves the search efficiency. Intuitively, this strategy first quickly approaches the manifold of adversarial inputs, and then searches for inputs satisfying both prediction and interpretation constraints.

Label Smoothing – Recall that we measure the prediction loss $\ell_{\text{prd}}(f(x), c_t)$ with cross entropy. When attacking GRAD, ADV^2 may generate intermediate inputs that cause f to make over-confident predictions (e.g., with probability 1). The all-zero gradient of ℓ_{prd} prevents the attack from finding inputs with desirable interpretations. To solve this, we refine cross entropy with label smoothing [55]. We sample y_{c_t} from a uniform distribution $\mathbb{U}(1-\rho,1)$ and define $y_c = \frac{1-y_{c_t}}{|C|-1}$ for $c \neq c_t$ and $\ell_{\text{prd}}(f(x),c_t) = -\sum_{c\in C} y_c \log f_c(x)$. During the attack, we gradually decrease ρ from 0.05 to 0.01.

Multistep Lookahead – In implementing Algorithm 1, we apply multiple steps of gradient descent in both updating m (line 3) and computing the surrogate map $m_*(x)$ (line 4), which is observed to lead to faster convergence in our empirical evaluation. Further, to improve the optimization stability, we use the average gradient to update m. Specifically, let $\{m_j^{(i)}\}$ be the sequence of maps obtained at the i-th iteration by applying multistep gradient descent. We use the aggregated interpretation loss $\sum_j \|m_j^{(i)} - m_t\|_2^2$ to compute the gradient for updating m.

Adaptive Learning Rate – To improve the convergence of Algorithm 1, we also dynamically adjust the learning rate for updating m and x. At each iteration, we use a running Adam optimizer as a meta-learner [4] to estimate the optimal learning rate for updating m (line 3). We update x in a two-step fashion to stabilize the training: (i) first updating x in terms of the prediction loss ℓ_{prd}, and (ii) updating it in terms of the interpretation loss ℓ_{int}. During (ii), we use a binary search to find the largest step size, such that x's confidence is still above a certain threshold κ after the perturbation.

Periodical Reset – Recall that in Algorithm 1, we update the estimate of the attribution map by following gradient descent on ℓ_{map}. As the number of update steps increases, this estimate may deviate significantly from the true map generated by the Mask interpreter, which negatively impacts the attack effectiveness. To address this, periodically (e.g., every 50 iterations), we replace the estimated map with the map $g(x;f)$ that is directly computed by Mask based on the current adversarial input. At the same time, we reset the Adam step parameter to correct its internal state.

4 Attack Evaluation

Next we conduct an empirical study of Adv^2 on a variety of DNNs and interpreters from both qualitative and quantitative perspectives. Specifically, our experiments are designed to answer the following key questions about Adv^2:

Q$_1$: Is it effective to deceive target classifiers?
Q$_2$: Is it effective to mislead target interpreters?
Q$_3$: Is it evasive with respect to attack detection methods?
Q$_4$: Is it effective in real security-critical applications?
Q$_5$: Is it flexible to adopt alternative attack frameworks?

Experimental Setting

We first introduce the setting of our empirical evaluation.

Datasets – Our evaluation primarily uses ImageNet [12], which consists of 1.2 million images from 1,000 classes. Every image is center-cropped to 224×224 pixels. For a given classifier f, from the validation set of ImageNet, we randomly sample 1,000 images that are classified correctly by f to form our test set. All the pixels are normalized to $[0,1]$.

Classifiers – We use two state-of-the-art DNNs as the classifiers, ResNet-50 [22] and DenseNet-169 [24], which respectively attain 77.15% and 77.92% top-1 accuracy on ImageNet. Using two DNNs of distinct capacities (50 layers versus 169 layers) and architectures (residual blocks versus dense blocks), we factor out the influence of the characteristics of individual DNNs.

Interpreters – We adopt Grad [50], Cam [64], Rts [10], and Mask [16] as the representatives of back-propagation-, representation-, model-, and perturbation-guided interpreters respectively. We adopt their open-source implementation in our evaluation. As Rts is tightly coupled with its target DNN (i.e., ResNet), we train a new masking model for DenseNet. To assess the validity of the implementation, we evaluate all the interpreters in a weakly semi-supervised localization task [8] using the benchmark dataset and method in [10]. Table 2 summarizes the results. The performance of all the interpreters is consistent with that reported in [10], with slight variation due to the difference of underlying DNNs.

Interpreter	Grad	Cam	Mask	Rts
Measure	43.1	43.8	45.2	34.2

Table 2. Performance of the interpreters in this paper in a weakly semi-supervised localization task (with ResNet as the classifier).

Attacks – We implement all the variants of Adv^2 in § 3 on the Pgd framework. In addition, we also implement Adv^2 on a spatial transformation framework (StAdv) [2]. We compare Adv^2 with regular Pgd [35], a universal first-order adversarial attack. For both Adv^2 and Pgd, we assume the setting of targeted attacks, in which the adversary attempts to force the DNNs to misclassify the adversarial inputs into randomly designated classes. The parameter settings of all the attacks are summarized in Appendix B.

Q1. Attack Effectiveness (Prediction)

We first evaluate the effectiveness of Adv^2 in terms of deceiving target DNNs. The effectiveness is measured using *attack success rate*, which is defined as

$$\text{Attack Success Rate (ASR)} = \frac{\#\text{successful trials}}{\#\text{total trials}}$$

and *misclassification confidence* (MC), which is the probability assigned by the DNN to the target class c_t.

	ResNet				DenseNet			
	Grad	Cam	Mask	Rts	Grad	Cam	Mask	Rts
P	100% (1.0)				100% (1.0)			
A	100% (0.99)	100% (1.0)	98% (0.99)	100% (1.0)	100% (0.98)	100% (1.0)	96% (0.98)	100% (1.0)

Table 3. Effectiveness of Pgd (P) and Adv^2 (A) against different classifiers and interpreters in terms of ASR (MC).

Table 3 summarizes the attack success rate and misclassification confidence of Adv^2 and Pgd against different combinations of classifiers and interpreters. Note that as Pgd is only applied on the classifier, its effectiveness is agnostic to the interpreters. To make fair comparison, we fix the maximum number of iterations as 1,000 for both attacks. It is observed that ADV^2 achieves high success rate (above 95%) and misclassification confidence (above 0.98) across all the cases, which is comparable with the regular PGD attack. We thus have the following conclusion.

Observation 1

Despite its dual objectives, ADV^2 is as effective as regular adversarial attacks in deceiving target DNNs.

Q2. Attack Effectiveness (Interpretation)

Next we evaluate the effectiveness of ADV^2 in terms of generating similar interpretations to benign inputs. Specifically, we compare the interpretations of benign and adversarial inputs, which is crucial for understanding the security implications of using interpretability as a means of defenses [13, 57]. Due to the lack of standard metrics for interpretation plausibility, we use a variety of measures in our evaluation.

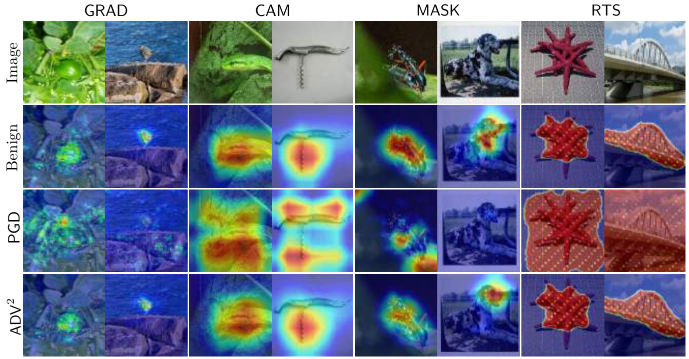


Figure 4: Attribution maps of benign and adversarial (PGD, ADV^2) inputs with respect to GRAD, CAM, MASK, and RTS on ResNet.

Visualization – We first qualitatively compare the interpretations of benign and adversarial (PGD, ADV^2) inputs. Figure 4 show a set of sample inputs and their attribution maps with respect to GRAD, CAM, MASK, and RTS (more samples in Appendix C1). Observe that in all the cases, the ADV^2 inputs generate interpretations perceptually indistinguishable from their benign counterparts. In comparison, the PGD inputs are easily identifiable by inspecting their attribution maps.

***L_p* Measure –** Besides qualitatively comparing the attribution maps of benign and adversarial inputs, we also measure their similarity quantitatively. By considering attribution maps as matrices, we measure the $\mathcal{L}_1$ distance between benign and adversarial maps. Figure 5 summarizes the results (other $\mathcal{L}_p$ measures in Appendix C1). For comparison, we normalize all the measures to $[0,1]$ by dividing them by the number of pixels.

We have the following observations. (i) Compared with PGD, ADV^2 generates attribution maps much more similar to benign cases. The average $\mathcal{L}_1$ measure of ADV^2 is more than 60% lower than PGD across all the interpreters. (ii) The effectiveness of ADV^2 varies with the target interpreter. For instance, compared with other interpreters, the difference between PGD and ADV^2 is relatively marginal on GRAD, implying that different interpreters may inherently feature varying robustness against ADV^2. (iii) The effectiveness of ADV^2 seems insensitive to the underlying DNN. On both ResNet and DenseNet, it achieves similar $\mathcal{L}_1$ measures.

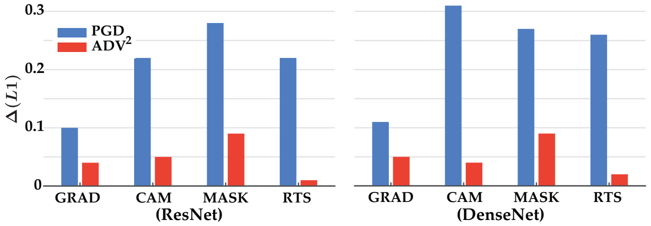


Figure 5: Average $\mathcal{L}_1$ distance between benign and adversarial (PGD, ADV^2) attribution maps.

IoU Test – Another quantitative measure for the similarity of attribution maps is the intersection-over-union (IoU) score. It is widely used in object detection [21] to compare model predictions with ground-truth bounding boxes. Formally, the IoU score of a binary-valued map m with respect to a baseline map $m_\circ$ is defined as their Jaccard similarity: $\text{IoU}(m) = |O(m) \cap O(m_\circ)|/|O(m) \cup O(m_\circ)|$, where $O(m)$ denotes the set of non-zero dimensions in m. In our case, as the values of attribution maps are floating numbers, we first apply threshold binarization on the maps.

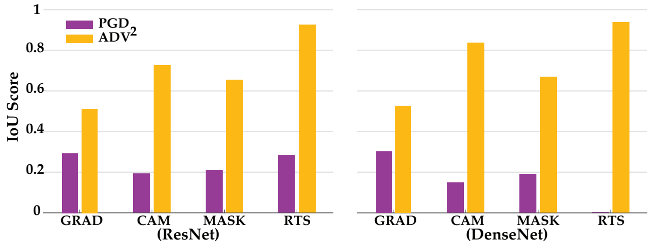


Figure 6: IoU scores of adversarial attribution maps (PGD, ADV^2) with respect to benign maps.

Following a typical rule used in the object detection task [21] where a detected region of interest (RoI) is considered positive if its IoU score is above 0.5 with respect to a ground-truth mask, we thus consider an attribution map as plausible if its IoU score exceeds 0.5 with respect to the benign attribution map. Figure 6 compares the average IoU scores of adversarial maps (PGD, ADV^2) with respect to the benign cases. Observe that ADV^2 achieves IoU scores above 0.5 across all the interpreters, which are more than 40% higher than PGD in all the cases. Especially on RTS, in which the attribution maps are natively binary-valued, ADV^2 achieves IoU scores above 0.9 on both ResNet and DenseNet.

Based on both qualitative and quantitative measures, we have the following conclusion.

Observation 2

ADV^2 is able to generate adversarial inputs with interpretations highly similar to benign cases.

Q3. Attack Evasiveness

Intuitively, from the adversary's perspective, ADV^2 entails a search space for adversarial inputs no larger than its underlying adversarial attack (e.g., PGD), as ADV^2 needs to optimize both the prediction loss ℓ_{prd} and interpretation loss ℓ_{int}, while ADV^2 only needs to optimize ℓ_{prd}. Next we compare PGD and ADV^2 in terms of their evasiveness with respect to adversarial attack detection methods.

Basic ADV^2 – To be succinct, we consider feature squeezing (FS) [61] as a concrete detection method. FS reduces the adversary's search space by coalescing inputs corresponding to different feature vectors into a single input, and detects adversarial inputs by comparing their predictions under original and squeezed settings. This operation is implemented in the form of a set of "squeezers": bit depth reduction, local smoothing, and non-local smoothing.

Squeezer	Setting	PGD	MASK-A	RTS-A	MASK-A*	RTS-A*
Bit Depth Reduction	2-bit	92.3%	84.1%	94.0%	11.7%	29.4%
	3-bit	72.7%	89.2%	88.3%	35.9%	13.9%
L. Smoothing	3×3	97.3%	98.6%	99.0%	16.5%	3.4%
N. Smoothing	11-3-4	52.3%	74.7%	75.3%	51.7%	29.4%

Table 4. Detectability of adversarial inputs by PGD, basic ADV^2 (A), and adaptive ADV^2 (A*) using feature squeezing.

Table 4 lists the detection rate of adversarial inputs (PGD, ADV^2) using different types of squeezers on ResNet. Observe that the squeezers seem effective to detect both ADV^2 and PGD inputs. For instance, local smoothing achieves higher than 97% success rate in detecting both ADV^2 and PGD inputs, with difference less than 2%. We thus have:

> Observation 3
>
> The overall detectability of ADV^2 and PGD with respect to feature squeezing is not significantly different.

Adaptive ADV^2 – We now adapt ADV^2 to evade the detection of FS. Related to existing adaptive attacks against FS [23], this optimization is interesting in its own right. Specifically, for smoothing squeezers, we augment the loss function $\ell_{\text{adv}}(x)$ (Eqn (3)) with the term $\ell_{\text{sqz}}(f(x), f(\psi(x)))$, which is the cross entropy of the predictions of original and squeezed inputs (ψ is the squeezer).

Algorithm 2: Adaptive ADV^2 against Feature Squeezing.

```
Input: $x_\circ$: benign input; $c_t$: target class; $f$: target DNN; $g$:
       target interpreter; $\psi$: bit depth reduction; $i$: bit depth
Output: $x_*$: adversarial input
// augmented $\ell_{\text{adv}}$ with $\ell_{\text{sqz}}$ w.r.t. smoothing
// attack in squeezed space
$x_+ \leftarrow$ PGD on $\psi(x_\circ)$ with target $c_t$ and $\alpha = 1/2^i$;
// attack in original space
search for $x_* = \arg\min_{x \in \mathcal{B}_\varepsilon(x_\circ)} \ell_{\text{adv}}(x) + \lambda \|f(x) - f(x_+)\|_1$;
return $x_*$;
```

For bit depth reduction, we use a two-stage strategy. (i) We first search in the squeezed space for an adversarial input x_+ that is close to $x_\circ$'s ε-neighborhood. To do so, we run PGD over $\psi(x_\circ)$ with learning rate $\alpha = 1/2^i$ (i is the bit depth). (ii) We then search in $x_\circ$'s ε-neighborhood for an adversarial input x_* that is classified similarly as x_+. To do so, we augment the loss function $\ell_{\text{adv}}(x)$ with a probability loss term $\|f(x) - f(x_+)\|_1$ ($f(x_+)$ is x_+'s probability vector), and then apply PGD to search for x_* within $x_\circ$'s ε-neighborhood. The overall algorithm is sketched in Algorithm 2.

Metric	MASK			RTS		
	P	A	A*	P	A	A*
$\Delta\mathcal{L}_1$	0.28	0.09	0.09	0.22	0.01	0.02
IoU	0.21	0.65	0.61	0.29	0.93	0.94

Table 5. $\mathcal{L}_1$ measures and IoU scores of adversarial attribution maps (PGD, basic and adaptive ADV^2) with respect to benign maps.

Table 4 summarizes the detection rate of adversarial inputs generated by adaptive ADV^2, which drops significantly, compared with the case of basic ADV^2. Note that here we only show the possibility of adapting ADV^2 to evade a representative detection method, and consider an in-depth study on this matter as our ongoing work. Meanwhile, we compare the $\mathcal{L}_1$ measures and IoU scores of the attribution maps generated by basic and adaptive ADV^2 (with respect to the benign maps). Table 5 shows the results. Observe that the optimization in adaptive ADV^2 has little impact on its attack effectiveness against the interpreters. We may thus conclude:

> Observation 4
>
> It is possible to adapt ADV^2 to generate adversarial inputs evasive with respect to feature squeezing.

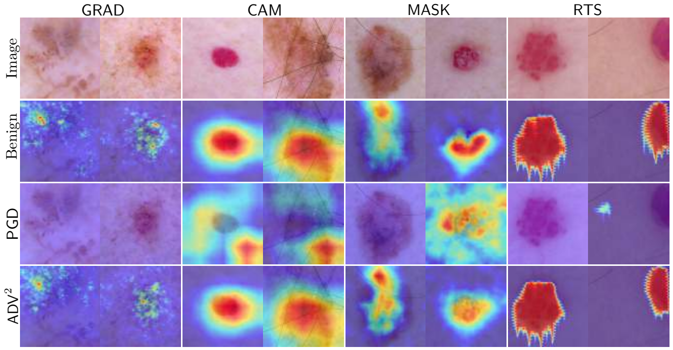


Figure 7: Attribution maps of benign and adversarial (ADV^2) inputs in the skin cancer screening application.

Q4. Real Application

We now evaluate the effectiveness of ADV^2 in real security-critical applications. We use the skin cancer screening task from the ISIC 2018 challenge [18] as a case study, in which given skin lesion images are categorized into a seven-disease taxonomy. We adopt a competition-winning model[1] (with ResNet as its backbone) as the classifier, which attains 82.27% weighted multi-class accuracy on the holdout set (more details in Appendix C2).

[1] https://github.com/ngessert/isic2018

We apply ADV^2 on this classifier and measure its effectiveness of generating plausible interpretations. Figure 7 shows a set of samples and their attribution maps on the four interpreters. Observe that ADV^2 generates interpretations visually indiscernible from their benign counterparts in all the cases.

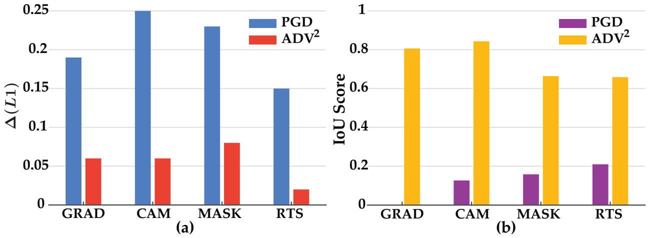


Figure 8: $\mathcal{L}_1$ measures (a) and IoU scores (b) of adversarial attribution maps (PGD, ADV^2) with respect to benign maps.

This similarity is further quantitatively validated in Figure 8, which shows the $\mathcal{L}_1$ measures (other $\mathcal{L}_\text{p}$ measures in Appendix C2) and IoU scores of the maps generated by ADV^2 with respect to the benign maps. For instance, the IoU scores of ADV^2 exceed 0.62 across all the interpreters.

Q5. Alternative Attack Framework

Besides the PGD framework, ADV^2 can also be flexibly built upon alternative frameworks. Here we construct ADV^2 upon STADV [60], a spatial transformation-based adversarial attack. The implementation details are given in Appendix A1.

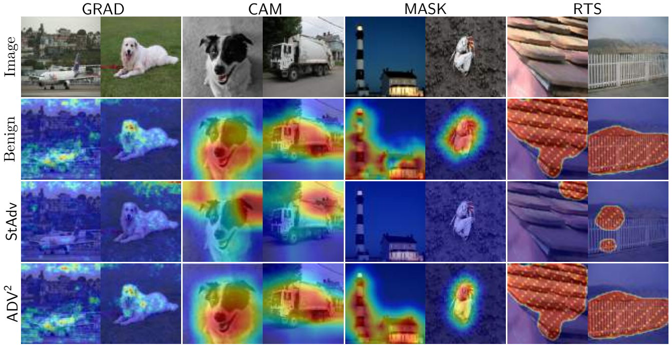


Figure 9: Attribution maps of benign and adversarial (STADV, STADV-based ADV^2) inputs with respect to GRAD, CAM, MASK, and RTS on ResNet.

Figure 9 visualizes sample benign and adversarial inputs and their interpretations. Compared with STADV, ADV^2 generates adversarial inputs with maps much more similar to the benign cases, highlighting the effectiveness of ADV^2 constructed upon the STADV framework.

This observation is also confirmed by the $\mathcal{L}_1$ measures and IoU scores of adversarial attribution maps, which are shown in Figure 10 (more results in Appendix C3). Interestingly, compared with the other interpreters, MASK seems more resilient to STADV-based ADV^2. The comparison with the results of PGD-based ADV^2 (Figure 5 and 6) implies that the (relative) robustness of different interpreters may vary with the concrete attacks (details in § 5).

Overall we have the following conclusion.

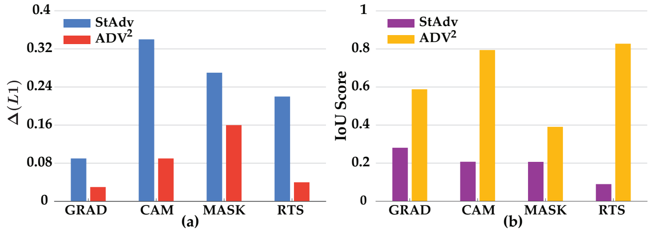


Figure 10: $\mathcal{L}_1$ measures (a) and IoU scores (b) of adversarial attribution maps (STADV, STADV-based ADV^2) with respect to benign maps on ResNet.

Observation 5

As a general class of attacks, ADV^2 can be flexibly built upon alternative adversarial attack frameworks.

5 Discussion

While it is shown in § 4 that ADV^2 is effective against a range of classifiers and interpreters, the cause of this effectiveness is unclear yet. Next we conduct a study on this root cause from both analytical and empirical perspectives. Based on our findings, we further discuss potential countermeasures against ADV^2.

Q1. Root of Attack Vulnerability

Recall that the formulation of ADV^2 in Eqn (3) defines two seemingly conflicting objectives: (i) maximizing the prediction change while (ii) minimizing the interpretation change. We thus conjecture that the effectiveness of ADV^2 may stem from the partial independence between a classifier and its interpreter – the interpreter's explanations only partially describe the classifier's predictions, making it practical to exploit both models simultaneously.

To validate the existence of this prediction-interpretation gap, we consider ADV^2 targeting randomly generated predictions and interpretations. For a given input $x_\circ$, we randomly generate a target class c_t and a target interpretation m_t, and search for an adversarial input x_* that triggers the classifier to misclassify it as c_t and also generates an interpretation similar to m_t (i.e., $f(x_*) = c_t$ and $g(x_*; f) \approx m_t$). Intuitively, if ADV^2 is able to find such x_*, it indicates that the classifier and its interpreter can be manipulated separately; in other words, they are only partially aligned with each other.

Random Patch Interpretation – In the first case, for a given input, we define its target attribution map by (i) sampling a patch of random shape (either a rectangle or a circle), random angle, and random position over the input, and (ii) setting the elements inside the patch as '1' and that outside it as '0'. Typically this target map deviates significantly from its benign counterpart, due to its randomness.

We evaluate the effectiveness of ADV^2 under this setting. Table 6 summarizes the attack success rate of ADV^2 on ResNet. Observe that compared with Table 3, targeting random patch interpretations has little impact on the attack ef-

	GRAD	CAM	MASK	RTS
ADV^2	100% (0.98)	100% (1.0)	99% (0.95)	100% (1.0)

Table 6. ASR (MC) of ADV^2 targeting random patch interpretations.

fectiveness in terms of deceiving the classifiers, implying that the space of adversarial inputs is sufficiently large to contain ones with targeted interpretations.

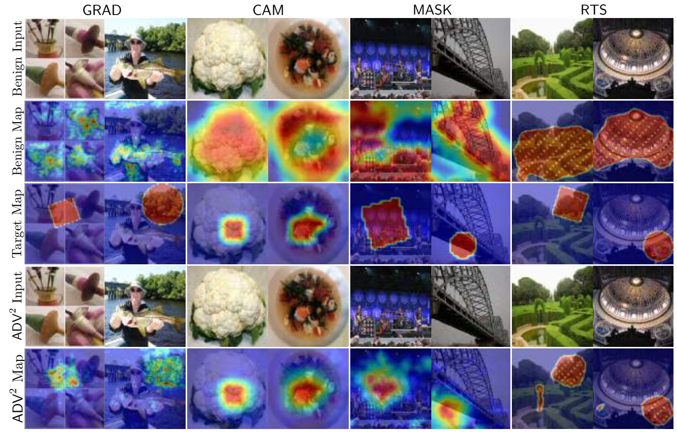


Figure 11: Visualization of ADV^2 targeting random patch interpretations across different interpreters on ResNet.

We then evaluate the effectiveness of ADV^2 in terms of generating the target interpretations. For a given benign input $x_\circ$ and a target random patch map m_t, ADV^2 attempts to generate an adversarial input c_t with the interpretation similar to m_t. Figure 11 visualizes a set of sample results. Note that in all the cases the ADV^2 maps appear visually similar to the target maps, highlighting the attack effectiveness. This effectiveness is further validated in Table 7. Observe that across all the interpreters, an ADV^2 map is much more similar to its target map, compared with its benign counterpart.

	GRAD	CAM	MASK	RTS
$\Delta_b \mathcal{L}_1$	0.16	0.50	0.42	0.49
$\Delta_t \mathcal{L}_1$	0.10	0.04	0.15	0.07

Table 7. Comparison of ADV^2 and target maps (Δ_t) and that of ADV^2 and benign maps (Δ_b), measured by $\mathcal{L}_1$ distance.

Random Class Interpretation – In the second case, for a given input (with c_t as the target class), we instantiate its target interpretation with the attribution map of a benign input randomly sampled from another class $\tilde{c}_t$. We particularly enforce $c_t \neq \tilde{c}_t$; in other words, the adversarial input is misclassified into one class but interpreted as another one.

	GRAD	CAM	MASK	RTS
ADV^2	100% (0.99)	100% (0.99)	100% (0.99)	100% (1.0)

Table 8. ASR (MC) of ADV^2 with random class interpretations.

The ASR of ADV^2 is summarized in Table 8. Observe that targeting random class interpretations has little influence on the attack effectiveness of deceiving the classifiers. Figure 12 visualizes a set of sample target and ADV^2 inputs and their interpretations (DenseNet results in Appendix C4). Note that the target and ADV^2 inputs are fairly distinct, but with highly similar interpretations. This is quantitatively validated by their $\mathcal{L}_1$ measures and IoU scores listed in Figure 13.

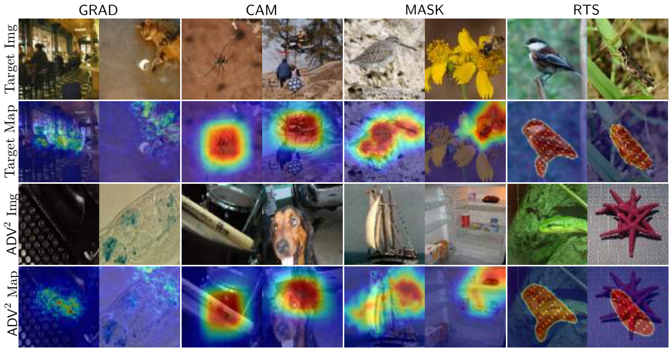


Figure 12: Target and adversarial (ADV^2) inputs and their attribution maps on ResNet.

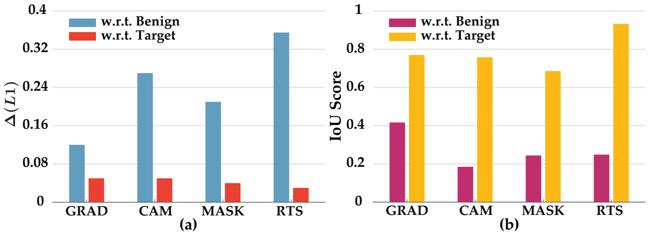


Figure 13: $\mathcal{L}_1$ measures (a) and IoU scores (b) of adversarial maps with respect to benign and target cases on ResNet.

The experiments above show that it is practical to generate adversarial inputs targeting arbitrary predictions and interpretations. We can therefore conclude:

> **Observation 6**
>
> A DNN and its interpreter are often not fully aligned, allowing the adversary to exploit both models simultaneously.

Q2. Root of Prediction-Interpretation Gap

Next we explore the fundamental causes of this prediction-interpretation gap. We speculate one following possible explanation as: existing interpretation models do not comprehensively capture the dynamics of DNNs, each only describing one aspect of their behavior.

Specifically, GRAD solely relies on the gradient information; MASK focuses on the input-prediction correspondence while ignoring the internal representations; CAM leverages the deep representations at intermediate layers, but neglecting the input-prediction correspondence; RTS uses the internal representations in an auxiliary encoder and the input-interpretation correspondence in the training data, which however may deviate from the true behavior of DNNs.

Intuitively the exclusive focus on one aspect (e.g., input-prediction correspondence) of the DNN behavior results in loose constraints: when performing the attack, the adversary only needs to ensure that benign and adversarial inputs cause DNNs to behave similarly from one specific perspective. We validate this speculation from two observations, low attack transferability and disparate attack robustness.

Attack Transferability – One intriguing property of adversarial inputs is their transferability: an adversarial input effective against one DNN is often found effective against another DNN, though it is not crafted on the second one [33, 36, 40]. In this set of experiments, we investigate whether such transferability exists in attacks against interpreters; that is, whether an adversarial input that generates a plausible interpretation against one interpreter is also able to generate a probable interpretation against another interpreter.

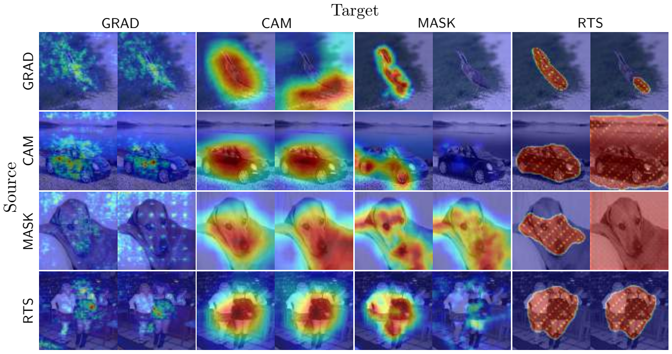


Figure 14: Visualization of attribution maps of adversarial inputs across different interpreters on ResNet.

Specifically, for each given interpreter g, we randomly select a set of adversarial inputs crafted against g (source) and compute their interpretations on another interpreter g' (target). Figure 14 illustrates the attribution maps of a given adversarial input on g and g'. Further, for each case, we compare the adversarial map (right) against the corresponding benign map (left). Observe that the interpretation transferability is fairly low: an adversarial input crafted against one interpreter g rarely generates highly plausible interpretation on another interpreter g'.

	GRAD	CAM	MASK	RTS
GRAD	0.04	0.24	0.22	0.24
CAM	0.09	0.05	0.18	0.13
MASK	0.12	0.34	0.09	0.74
RTS	0.10	0.17	0.20	0.01
PGD	0.10	0.22	0.28	0.22

Table 9. $\mathcal{L}_1$ distance between attribution maps of adversarial (ADV^2, PGD) on ResNet (row/column as source/target).

We further quantitatively validate this observation. Table 9 measures the $\mathcal{L}_1$ distance between the adversarial and benign attribution maps across different interpreters. For comparison, it also shows the $\mathcal{L}_1$ measure for the adversarial inputs generated by PGD. Observe that the adversarial inputs crafted on g tends to generate low-quality interpretations on a different interpreter g', with quality comparable to that generated by an interpretation-agnostic attack (i.e., PGD). We can therefore conclude:

Observation 7

The transferability of adversarial inputs across different interpreters seems low.

Attack Robustness – It is observed in § 4 that the effectiveness of ADV^2 varies with the target interpreter. As shown in Figure 6, among all the interpreters, ADV^2 attains the lowest IoU scores on GRAD, suggesting that GRAD may be more robust against ADV^2. This observation may be explained as follows: GRAD uses the gradient magnitude of each input feature to measure its relevance to the model prediction; meanwhile, ADV^2 heavily uses the gradient information to optimize the prediction loss ℓ_{prd}; it is inherently difficult to minimize ℓ_{prd} while keeping the gradient intact.

We validate the conjecture by analyzing the robustness of integrated gradient (IG) [53], another back-propagation-guided interpreter, against ADV^2. Due to their fundamental equivalence [3], the discussion here also generalizes to other back-propagation interpreters (e.g., [48, 50, 51]).

At a high level, for the i-th feature of a given input x, IG computes its attribution $m[i]$ by aggregating the gradient of $f(x)$ along the path from a baseline input $\bar{x}$ to x:

$$m[i] = (x[i] - \bar{x}[i]) \int_0^1 \frac{\partial f(tx + (1-t)\bar{x})}{\partial x[i]} \mathrm{d}t \tag{11}$$

Like other back-propagation interpretation models [3], IG satisfies the desirable completeness axiom [48] that the attributions sum up to the difference between f's predictions for the given input x and the baseline $\bar{x}$.

To simplify the exposition, let us assume a binary classification setting with classes $\mathcal{C} = \{+, -\}$. The DNN f predicts the probability of x belonging to the positive class as $f(x)$. Given an input $x_\circ$ from the negative class, the adversary attempts to craft an adversarial input x_* to force f to misclassify x_* as positive. We define the prediction loss as $\ell_{\text{prd}}(x_*) = f(x_*) - f(x_\circ)$ (i.e., the increase in the probability of positive prediction), which can be computed as:

$$\ell_{\text{prd}}(x_*) = \int_0^1 \nabla f(tx_* + (1-t)x_\circ)^\top (x_* - x_\circ) \mathrm{d}t \tag{12}$$

Meanwhile, we define the interpretation loss as $\ell_{\text{int}}(x_*) = \|m_\circ - m_*\|_1$, where $m_\circ$ and m_* are the attribution maps of $x_\circ$ and x_* respectively. While it is difficult to directly quantify $\ell_{\text{int}}(x_*)$, we may use the attribution map of x_* with $x_\circ$ as a surrogate baseline:

$$\Delta m[i] = (x_*[i] - x_\circ[i]) \int_0^1 \frac{\partial f(tx_* + (1-t)x_\circ)}{\partial x_*[i]} \mathrm{d}t \tag{13}$$

which quantifies the impact of the i-th input feature on the difference of $f(x_\circ)$ and $f(x_*)$. Thus, $\ell_{\text{int}}(x_*) = \|\Delta m\|_1$.

Proposition 1. *With* IG*, the prediction loss is upper bounded by the interpretation loss as:* $\ell_{\text{prd}}(x_*) \leq \ell_{\text{int}}(x_*)$.

Proof. We define u as the input difference $u = (x_* - x_\circ)$ and v as the integral vector with its i-th element $v[i]$ defined as

$$v[i] = \int_0^1 \frac{\partial f(tx_* + (1-t)x_\circ)}{\partial x_*[i]} \mathrm{d}t$$

According to the definitions, we have $\ell_{\text{prd}}(x_*) = u^\top v$ and $\ell_{\text{int}}(x_*) = \|u \odot v\|_1$, where $\odot$ is the Hadamard product.

We have the following derivation: $\ell_{\text{prd}}(x_*) = \sum_i u[i]v[i] \leq \sum_i \|u[i] \cdot v[i]\| = \ell_{\text{int}}(x_*)$. Thus the prediction loss is upper-bounded by the interpretation loss. □

In other words, in order to force x_* to be misclassified with high confidence, the difference of benign and adversarial attribution maps needs to be large. As the objectives of ADV^2 here is to maximize the prediction loss while minimizing the interpretation loss. The coupling between prediction and interpretation losses results in a fundamental conflict.

Note that however this conflict does not preclude effective adversarial attacks. First, the constraint of prediction and interpretation losses may be loose. Let γ_{prd} and γ_{int} be the thresholds of effective attacks. That is, for an effective attack, $\ell_{\text{prd}}(x_*) \geq \gamma_{\text{prd}}$ and $\ell_{\text{int}}(x_*) \leq \gamma_{\text{int}}$. There could be cases that $\gamma_{\text{prd}} \ll \gamma_{\text{int}}$, making ADV^2 still highly effective (e.g., Figure 8). Second, the adversary may pursue attacks that rely less on the gradient information to circumvent this conflict.

Overall, with the evidence of low attack transferability and disparate attack robustness, we can conclude:

Observation 8

Existing interpreters tend to focus on distinct aspects of DNN behavior, which may result in the prediction-interpretation gap.

Q3. Potential Countermeasures

Based on our findings, next we discuss potential countermeasures against ADV^2 attacks.

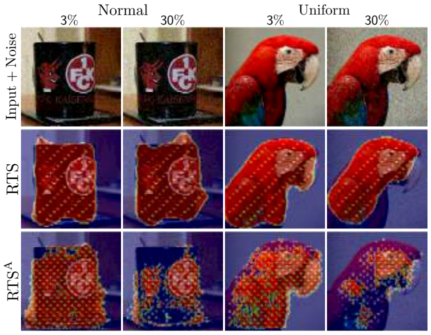
Defense 1: Ensemble Interpretation – Motivated by the observation that different interpreters focus on distinct aspects of DNN behavior (e.g., CAM focuses on deep representations while MASK focuses on input-prediction correspondence), a promising direction to defend against ADV^2 is to deploy multiple, complementary interpreters to provide a holistic view of DNN behavior.

Yet, two major challenges remain to be addressed. First, different interpreters may provide disparate interpretations (e.g., Figure 14). It is challenging to optimally aggregate such interpretations to detect ADV^2. Second, the adversary may adapt ADV^2 to the ensemble interpreter (e.g. optimizing the interpretation loss with respect to all the interpreters). It is crucial to account for such adaptiveness in designing the ensemble interpreter. We consider developing the ensemble defenses and exploring the adversary's adaptive strategies as our ongoing research directions.

Defense 2: Adversarial Interpretation – Along the second direction, we explore the idea of adversarial training. Recall that ADV^2 exploit the prediction-interpretation gap to generate adversarial inputs. Here we employ ADV^2 as a drive to minimize this gap during training interpreters.

Specifically, we propose an *adversarial interpretation distillation* (AID) framework. Let $\mathcal{A}$ be the ADV^2 attack. During training an interpreter g, for a given input $x_\circ$, besides the regular loss $\ell_{\text{map}}(x_\circ)$, we consider an additional loss term $\ell_{\text{aid}}(x_\circ) = -\|g(x_\circ) - g(\mathcal{A}(x_\circ))\|_1$, which is the negative $\mathcal{L}_1$ measure between the attribution maps of $x_\circ$ and its adversarial counterpart $\mathcal{A}(x_\circ)$. We encourage g to minimize this loss during the training (details in Appendix A2).

To assess the effectiveness of AID to reduce the prediction-interpretation gap, we use RTS as a concrete case study. Recall that RTS is a model-guided interpreter which directly predicts the interpretation of a given input. We construct two variants of RTS, a regular one and another with AID training (denoted by RTS$^{\mathtt{A}}$). We measure the sensitivity of the two interpreters to the underlying DNN behavior.



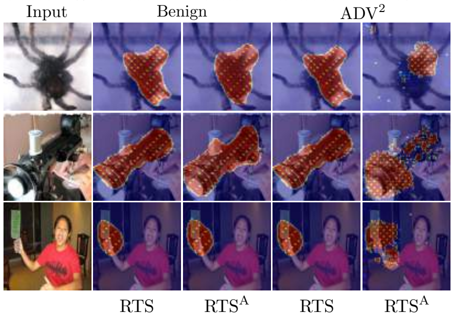
$\mathcal{L}_1$ measures

Noise	$\mathbb{N}$	$\mathbb{U}$
0%	0.01	0.01
	0.01	0.01
3%	0.01	0.01
	0.02	0.03
30%	0.09	0.10
	0.17	0.17

□ RTS ▨ RTS$^{\text{A}}$

Figure 15: Attribution maps generated by RTS and RTS$^{\mathtt{A}}$ under different noise levels and types (normal $\mathbb{N}$, uniform $\mathbb{U}$) on ResNet.

In the first case, we inject random noise (either normal or uniform) to the inputs and compare the attribution maps generated by the two interpreters. We consider two noise levels, which respectively cause 3% and 30% misclassification on the test set. Figure 15 shows a set of misclassified samples under the two noise levels. Observe that compared with RTS, RTS$^{\mathtt{A}}$ appears much more sensitive to the DNN's behavior change, by generating highly contrastive maps. This sensitivity is also quantitatively confirmed by the $\mathcal{L}_1$ measures between clean and noisy maps on RTS and RTS$^{\mathtt{A}}$. The findings also corroborate a similar phenomenon observed in [59]: the representations generated by robust models tend to align better with salient data characteristics.



	RTS	RTS$^{\mathtt{A}}$
Benign	\	0.03
ADV^2	0.01	0.10

$\mathcal{L}_1$ measures

Figure 16: Attribution maps of benign and adversarial (ADV^2) inputs with respect to RTS and RTS$^{\mathtt{A}}$ on ResNet.

In the second case, we assess the resilience of RTS$^{\mathtt{A}}$ against

ADV^2. In Figure 16, we compare the attribution maps of benign and adversarial inputs on RTS and RTS^A. It is observed that while ADV^2 generates adversarial inputs with interpretations fairly similar to benign cases on RTS, it fails to do so on RTS^A: the maps of adversarial inputs are fairly distinguishable from their benign counterparts. Moreover, RTS^A behaves almost identically to RTS on benign inputs, indicating that the AID training has little impact on benign cases. These findings are confirmed by the $\mathcal{L}_1$ measures as well.

Overall we have the following conclusion.

Observation 9

It is possible to exploit ADV^2 to reduce the prediction-interpretation gap during training interpreters.

6 Related Work

In this section, we survey three categories of work relevant to this work, namely, adversarial attacks and defenses, transferability, and interpretability.

Attacks and Defenses – Due to their widespread use in security-critical domains, machine learning models are increasingly becoming the targets of malicious attacks. Two primary threat models are considered in literature. Poisoning attacks – the adversary pollutes the training data to eventually compromise the target models [7]; Evasion attacks – the adversary manipulates the input data during inference to trigger target models to misbehave [11].

Compared with simple models (e.g., support vector machines), securing deep neural networks (DNNs) in adversarial settings entails more challenges due to their significantly higher model complexity [29]. One line of work focuses on developing new evasion attacks against DNNs [19, 35, 56]. Another line of work attempts to improve DNN resilience against such attacks by inventing new training and inference strategies [34, 42, 61]. Yet, such defenses are often circumvented by more powerful attacks [9] or adaptively engineered adversarial inputs [5], resulting in a constant arms race between attackers and defenders [31].

This work is among the first to explore attacks against DNNs with interpretability as a means of defense.

Transferability – One intriguing property of adversarial attacks is their transferability [56]: adversarial inputs crafted against one DNN is often effective against another one. This property enables black-box attacks – the adversary generates adversarial inputs based on a surrogate DNN and then applies them on the target model [33, 40]. To defend against such attacks, the method of ensemble adversarial training [58] has been proposed, which trains DNNs using data augmented with adversarial inputs crafted on other models.

This work complements this line of work by investigating the transferability of adversarial inputs across different interpretation models.

Interpretability – A plethora of interpretation models have been proposed to provide interpretability for black-box DNNs, using techniques based on back-propagation [50, 52, 53], intermediate representations [14, 47, 64], input perturbation [16], and meta models [10].

The improved interpretability is believed to offer a sense of security by involving human in the decision-making process. Existing work has exploited interpretability to debug DNNs [39], digest security analysis results [20], and detect adversarial inputs [32, 57]. Intuitively, as adversarial inputs cause unexpected DNN behaviors, the interpretation of DNN dynamics is expected to differ significantly between benign and adversarial inputs.

However, recent work empirically shows that some interpretation models seem insensitive to either DNNs or data generation processes [1], while transformation with no effect on DNNs (e.g., constant shift) may significantly affect the behaviors of interpretation models [27].

This work shows the possibility of deceiving DNNs and their coupled interpretation models simultaneously, implying that the improved interpretability only provides limited security assurance, which also complements prior work by examining the reliability of existing interpretation models from the perspective of adversarial vulnerability.

7 Conclusion

This work represents a systematic study on the security of interpretable deep learning systems (IDLSes). We present ADV^2, a general class of attacks that generate adversarial inputs not only misleading target DNNs but also deceiving their coupled interpretation models. Through extensive empirical evaluation, we show the effectiveness of ADV^2 against a range of DNNs and interpretation models, implying that the interpretability of existing IDLSes may merely offer a false sense of security. We identify the prediction-interpretation gap as one possible cause of this vulnerability, raising the critical concern about the current assessment metrics of interpretation models. Further, we discuss potential countermeasures against ADV^2, which sheds light on designing and operating IDLSes in a more robust and informative fashion.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1846151 and 1910546. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Shouling Ji is partially supported by NSFC under No. 61772466 and U1836202, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, and the Provincial Key Research and Development Program of Zhejiang, China under No. 2017C01055. Xiapu Luo is partially supported by Hong Kong RGC Project (No. PolyU 152279/16E, CityU C1008-16G).

References

[1] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity Checks for Saliency Maps. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2018.

[2] Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. ADef: an Iterative Algorithm to Construct Adversarial Deformations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[3] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross. Towards Better Understanding of Gradient-based Attribution Methods for Deep Neural Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[4] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2016.

[5] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE*, 10(7):e0130140, 2015.

[7] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning Attacks against Support Vector Machines. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2012.

[8] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, Deva Ramanan, and Thomas S Huang. Look and Think Twice: Capturing Top-Down Visual Attention with Feedback Convolutional Neural Networks. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015.

[9] Nicholas Carlini and David A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2017.

[10] P. Dabkowski and Y. Gal. Real Time Image Saliency for Black Box Classifiers. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[11] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial Classification. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

[12] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[13] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. Towards Explanation of DNN-based Prediction with Guided Feature Inversion. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[14] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. Towards Explanation of DNN-based Prediction with Guided Feature Inversion. *ArXiv e-prints*, 2018.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2017.

[16] Ruth C Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.

[17] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2018.

[18] Nils Gessert, Thilo Sentker, Frederic Madesta, Rüdiger Schmitz, Helge Kniep, Ivo M. Baltruschat, René Werner, and Alexander Schlaefer. Skin lesion diagnosis using ensembles, unscaled multi-crop evaluation and loss weighting. *ArXiv e-prints*, abs/1808.01694, 2018.

[19] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[20] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. LEMNA: Explaining Deep Learning Based Security Applications. In *Proceedings of ACM SAC Conference on Computer and Communications (CCS)*, 2018.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. In *USENIX Workshop on Offensive Technologies*, 2017.

[24] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2015.

[26] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and Understanding Recurrent Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016.

[27] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (Un)reliability of Saliency Methods. *ArXiv e-prints*, 2017.

[28] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[29] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[30] Min Lin, Qiang Chen, and Shuicheng Yan. Network in Network. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[31] X. Ling, S. Ji, J. Zou, J. Wang, C. Wu, B. Li, and T. Wang. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2019.

[32] Ninghao Liu, Hongxia Yang, and Xia Hu. Adversarial Detection with Model Interpretation. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[33] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-Box Attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[34] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[35] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[36] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal Adversarial Perturbations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] W. J. Murdoch, P. J. Liu, and B. Yu. Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[38] W. J. Murdoch and A. Szlam. Automatic Rule Extraction from Long Short Term Memory Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[39] A. Nguyen, J. Yosinski, and J. Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[40] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-box Attacks Using Adversarial Samples. *ArXiv e-prints*, 2016.

[41] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *Proceedings of ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*, 2017.

[42] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2016.

[43] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[44] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[45] S. Sabour, N. Frosst, and G. E Hinton. Dynamic Routing Between Capsules. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[46] F.M. Scherer. Heinrich von Stackelberg's Marktform und Gleichgewicht. *Journal of Economic Studies*, 23(5/6):58–70, 1996.

[47] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017.

[48] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2017.

[49] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, (7587):484–489, 2016.

[50] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[51] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. SmoothGrad: Removing Noise by Adding Noise. In *International Conference on Machine Learning Workshop*, 2017.

[52] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[53] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. In *Proceedings of IEEE Conference on Machine Learning (ICML)*, 2017.

[54] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2014.

[55] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[56] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[57] Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. Attacks Meet Interpretability: Attribute-Steered Detection of Adversarial Samples. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2018.

[58] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[59] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness May Be at Odds with Accuracy. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[60] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. Spatially Transformed Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

[61] W. Xu, D. Evans, and Y. Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

[62] Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Interpretable Convolutional Neural Networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[63] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable Convolutional Neural Networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[64] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Appendix

A. Implementation Details

A1: Details of StAdv-based ADV^2

We first briefly introduce the concept of spatial transformation. Let $\tilde{x}_i$ be the i-th pixel of adversarial input $\tilde{x}$ and $(\tilde{u}_i, \tilde{v}_i)$ be its spatial coordinates. With flow-based transformation, $\tilde{x}$ is generated from another input x by a per-pixel flow vector r, where $r_i = (\Delta u_i, \Delta v_i)$. The corresponding coordinates of $\tilde{x}_i$ in x are given by $(u_i, v_i) = (\tilde{u}_i + \Delta u_i, \tilde{v}_i + \Delta v_i)$. As (u_i, v_i) do not necessarily lie on the integer grid, bilinear interpolation [25] is used to compute $\tilde{x}_i$:

$$\tilde{x}_i = \sum_j x_j \max(0, 1 - |\tilde{u}_i + \Delta u_i - u_j|) \max(0, 1 - |\tilde{v}_i + \Delta v_i - v_j|)$$

where j iterates over the pixels adjacent to (u_i, v_i) in x. With STADV as the underlying attack framework, ADV^2 can be constructed as optimizing the following objective:

$$\min_r \ell_{\text{prd}}(f(x+r), c_t) + \lambda \ell_{\text{int}}(g(x+r; f), m_t) + \tau \ell_{\text{flow}}(r) \quad (14)$$

where $\ell_{\text{flow}}(r) = \sum_i \sum_{j \in \mathcal{N}(i)} \sqrt{\|\Delta u_i - \Delta u_j\|_2^2 + \|\Delta v_i - \Delta v_j\|_2^2}$ measures the magnitude of spatial transformation and τ is a hyper-parameter controlling its importance. In implementation, we solve Eqn (14) using an Adam optimizer.

A2: Details of AID

We use RTS as a concrete example to show the implementation of AID. In RTS, one trains a DNN g (parameterized by θ) to directly predict the attribution map $g(x; \theta)$ for a given input x. To train g, one minimizes the interpretation loss:

$$\ell_{\text{int}}(\theta) \triangleq \lambda_1 r_{\text{tv}}(g(x;\theta)) + \lambda_2 r_{\text{av}}(g(x;\theta)) - \log\left(f_c\left(\phi(x; g(x;\theta))\right)\right) + \lambda_3 f_c\left(\phi(x; 1 - g(x;\theta))\right)^{\lambda_4} \quad (15)$$

with all the terms defined similarly as in Eqn (8).

In AID, let $\mathcal{A}$ denote the ADV^2 attack. We further consider an adversarial distillation loss:

$$\ell_{\text{aid}}(\theta) \triangleq -\|g(x;\theta) - g(\mathcal{A}(x);\theta)\|_1 \quad (16)$$

which measures the difference of attribution maps of benign and adversarial inputs under the current interpreter $g(\cdot;\theta)$.

AID trains g by alternating between minimizing $\ell_{\text{int}}(\theta)$ and minimizing $\ell_{\text{aid}}(\theta)$ until convergence.

B. Parameter Setting

Here we summarize the default parameter setting for the attacks implemented in this paper.

B1. PGD-based ADV^2

For regular PGD, we set the learning rate $\alpha = 1./255$ and the perturbation threshold $\epsilon = 0.031$. Table 10 list the parameter setting of PGD-based ADV^2.

	Parameter	ResNet	DenseNet
GRAD	# iterations n_{total}	800	800
	ℓ_{int} coefficient (λ)	0.007	0.007
CAM	# iterations n_{total}	1200	1200
	ℓ_{int} coefficient (λ)	0.204/0.02	0.204
MASK	# iterations n_{total}	1000	1000
	# gradient descent steps n_{step}	4	4
	# iterations per reset n_{reset}	50	50
	max. search step size $\alpha_{\max}$	0.08	0.08
	# max. search steps n_{bs}	12	12
RTS	# iterations n_{total}	1200	1200
	ℓ_{int} coefficient (λ_1)	0.002/0.006	0.008
	ℓ_{prd} coefficient (λ_2)	0.1/-	0.1

Table 10. Parameter setting of PGD-based ADV^2. The setting for Q4 in § 4 (if different) is shown after '/'.

	Parameter	ResNet	DenseNet
GRAD	# iterations n_{total}	800	800
	ℓ_{int} coefficient (λ)	0.023	0.023
	ℓ_{flow} coefficient (τ)	0.0005	0.0005
CAM	# iterations n_{total}	600	600
	ℓ_{int} coefficient (λ)	0.653	0.653
	ℓ_{flow} coefficient (τ)	0.0005	0.0005
MASK	# iterations n_{total}	1000	1000
	# gradient descent steps n_{step}	4	4
	# iterations per reset n_{reset}	50	50
	ℓ_{int} coefficient (λ)	500	500
	ℓ_{flow} coefficient (τ)	0.004	0.005
RTS	# iterations n_{total}	600	600
	ℓ_{int} coefficient (λ_1)	0.0408	0.0612
	ℓ_{prd} coefficient (λ_2)	0.1	0.1
	ℓ_{flow} coefficient (τ)	0.0005	0.0005

Table 11. Parameter setting of STADV-based ADV^2.

B2. StAdv-based ADV^2

Table 11 list the parameter setting of STADV-based ADV^2.

The Adam optimizer in our experiments uses the hyper-parameter setting of $(\alpha, \beta_1, \beta_2) = (0.01, 0.9, 0.999)$.

C. Additional Experimental Results

Below we include more experimental results that complement the ones presented in § 4 and § 5.

C1. § 4 Q2 Attack Effectiveness (Interpretation)

Figure 17 shows a set of sample inputs (benign and adversarial) and their attribution maps generated by GRAD, CAM, MASK, and RTS on DenseNet.

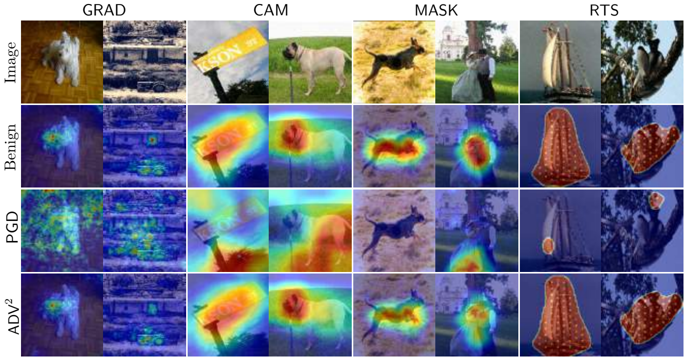


Figure 17: Attribution maps of benign and adversarial (PGD, ADV^2) inputs with respect to GRAD, CAM, MASK, and RTS on DenseNet.

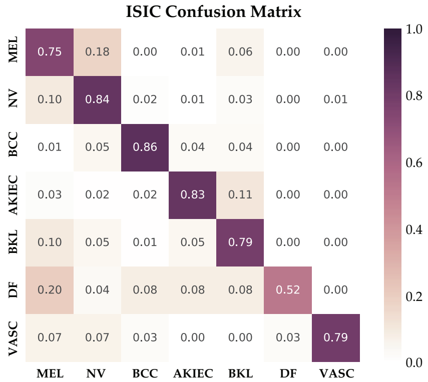
Table 12 lists the average $\mathcal{L}_p$ distance ($p = 1, 2$) between the attribution maps of benign and adversarial (PGD, ADV^2) inputs, which complements the results in Figure 5. We normalize the $\mathcal{L}_2$ measures by dividing them by the square root of the number of pixels.

	Attack	ResNet		DenseNet	
		$\Delta(\mathcal{L}_1)$	$\Delta(\mathcal{L}_2)$	$\Delta(\mathcal{L}_1)$	$\Delta(\mathcal{L}_2)$
GRAD	P	0.10	0.14	0.11	0.15
	A	0.04	0.07	0.05	0.07
CAM	P	0.22	0.28	0.31	0.36
	A	0.05	0.06	0.04	0.05
MASK	P	0.28	0.38	0.27	0.37
	A	0.09	0.16	0.09	0.17
RTS	P	0.22	0.42	0.26	0.48
	A	0.01	0.06	0.02	0.09

Table 12. $\mathcal{L}_p$ distance between attribution maps of benign and adversarial (P-PGD, A-ADV^2) inputs.

C2. § 4 Q4 Real Application

In § 4, we use the dataset from the ISIC 2018 challenge[2], and adopt a competition-winning model [18] (with ResNet as its backbone), which achieves the second place in the challenge. The confusion matrix in Figure 18 shows the performance of the classifier in our study.



	MEL	NV	BCC	AKIEC	BKL	DF	VASC
MEL	0.75	0.18	0.00	0.01	0.06	0.00	0.00
NV	0.10	0.84	0.02	0.01	0.03	0.00	0.01
BCC	0.01	0.05	0.86	0.04	0.04	0.00	0.00
AKIEC	0.03	0.02	0.02	0.83	0.11	0.00	0.00
BKL	0.10	0.05	0.01	0.05	0.79	0.00	0.00
DF	0.20	0.04	0.08	0.08	0.08	0.52	0.00
VASC	0.07	0.07	0.03	0.00	0.00	0.03	0.79

Figure 18: Confusion matrix of the classifier used in § 4 Q4 on the ISIC 2018 challenge dataset [18].

Table 13 lists the average $\mathcal{L}_p$ distance ($p = 1,2$) between the attribution maps of benign and adversarial (PGD, ADV^2) inputs in the case study of skin cancer diagnosis.

	Attack	$\Delta(\mathcal{L}_1)$	$\Delta(\mathcal{L}_2)$
GRAD	P	0.19	0.23
	A	0.06	0.09
CAM	P	0.25	0.31
	A	0.06	0.08
MASK	P	0.23	0.30
	A	0.08	0.11
RTS	P	0.15	0.26
	A	0.02	0.07

Table 13. $\mathcal{L}_p$ distance of attribution maps of benign and adversarial (PGD, ADV^2) inputs in the case study of skin cancer diagnosis.

C3. § 4 Q5 Alternative Attack Framework

Figure 19 visualizes attribution maps of benign and adversarial (STADV, STADV-based ADV^2) inputs on DenseNet. Figure 20 further compares the $\mathcal{L}_1$ measures and IoU scores (w.r.t. benign cases) of adversarial inputs on DenseNet.

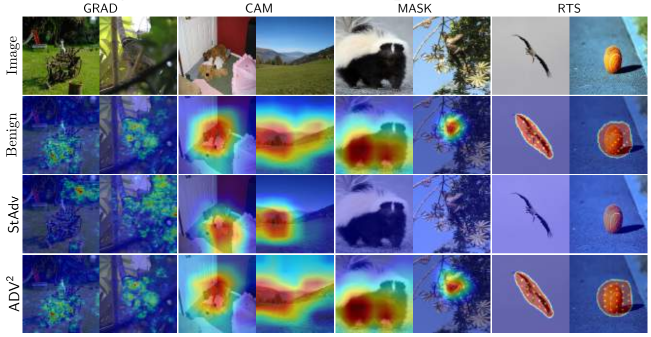


Figure 19: Attribution maps of benign and adversarial (STADV and STADV-based ADV^2) inputs on DenseNet.

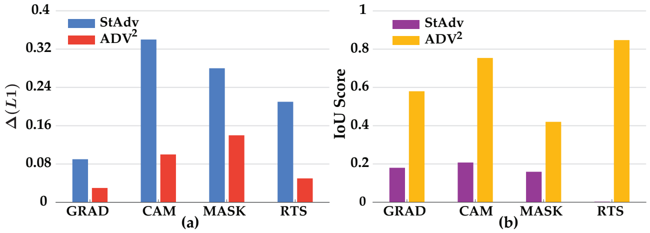


Figure 20: $\mathcal{L}_1$ measures and IoU scores of adversarial (STADV, STADV-based ADV^2) inputs w.r.t. benign maps on DenseNet.

C4. § 5 Q1 Random Class Interpretation

Figure 21 visualizes attribution maps of target and adversarial (ADV^2) inputs on DenseNet, which complements the results shown in Figure 12. Figure 22 compares the $\mathcal{L}_1$ measures and IoU scores of adversarial maps w.r.t. benign and target cases on DenseNet.

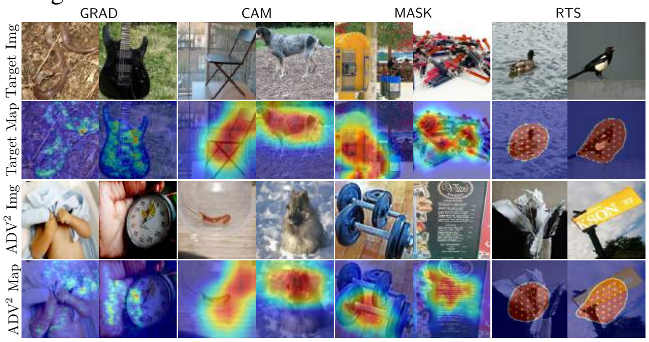


Figure 21: Target and adversarial (ADV^2) inputs and their attribution maps on DenseNet.

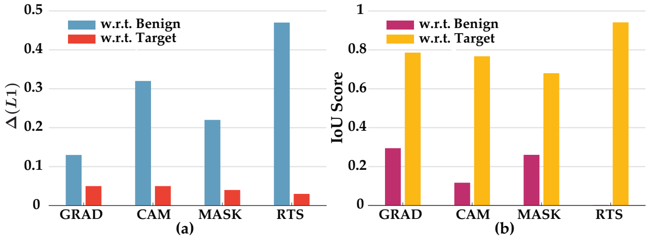


Figure 22: $\mathcal{L}_1$ measures (a) and IoU scores (b) of adversarial maps with respect to benign and target cases on DenseNet.

[2] https://challenge2018.isic-archive.com/task3/