Hardening Interpretable Deep Learning Systems: Investigating Adversarial Threats and Defenses

Eldor Abdukhamidov, Mohammed Abuhamad, Simon S. Woo, Eric Chan-Tin, and Tamer Abuhmed

***Abstract*—Deep learning methods have gained increasing attention in various applications due to their outstanding performance. For exploring how this high performance relates to the proper use of data artifacts and the accurate problem formulation of a given task, interpretation models have become a crucial component in developing deep learning-based systems. Interpretation models enable the understanding of the inner workings of deep learning models and offer a sense of security in detecting the misuse of artifacts in the input data. Similar to prediction models, interpretation models are also susceptible to adversarial inputs. This work introduces two attacks, AdvEdge and AdvEdge^{+}, which deceive both the target deep learning model and the coupled interpretation model. We assess the effectiveness of proposed attacks against four deep learning model architectures coupled with four interpretation models that represent different categories of interpretation models. Our experiments include the implementation of attacks using various attack frameworks. We also explore the attack resilience against three general defense mechanisms and potential countermeasures. Our analysis shows the effectiveness of our attacks in terms of deceiving the deep learning models and their interpreters, and highlights insights to improve and circumvent the attacks.**

***Index Terms*—Adversarial images, deep learning, security, transferability, interpretability.**

I. Introduction

DEEP Neural Networks (DNNs) have been increasingly incorporated into a wide range of applications due to their high predictive accuracy in various machine learning tasks, including image classification [2], and natural language processing [3]. Despite these extraordinary achievements, it is not yet entirely evident how DNN models make certain decisions due to their complex network architecture. Additionally, the susceptibility to adversarial manipulations is another shortcoming of DNN models. For example, adversarial examples, i.e., maliciously crafted inputs, can lead to unexpected model behavior during decision making.

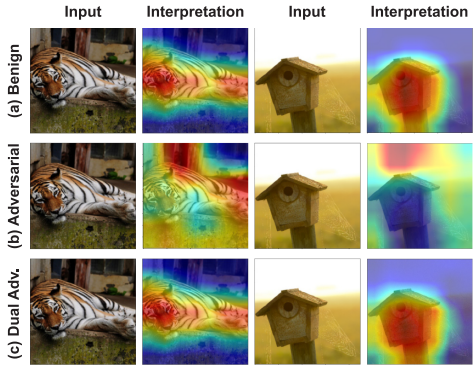

Fig. 1. Examples of (a) benign, (b) standard adversarial, and (c) dual adversarial images together with their interpretations based on the ResNet-50 model and CAM interpreter.

Numerous studies have been conducted to enhance the trustworthiness, reliability, and transparency of DNNs by providing interpretability at different levels (e.g., overall model [4], [5] and instance levels [6], [7]). Generally, interpretation is believed to aid in understanding the internal structure of models to debug them and identify possible manipulations. For example, Fig. 1(a) shows the causal relationship of attribution maps and images based on the prediction output. Fig. 1(b) shows successful adversarial examples that produce interpretation maps different from benign ones.

In the workflow of an interpretable deep learning system, a classifier (DNN model) and its coupled interpreter (interpretation model) construct interpretable deep learning systems (IDLS). IDLSes offer additional reliability and security in human-assisted decision-making processes, since experts can determine if an interpretation map fits the prediction of the DNN models. However, interpretability is also vulnerable to adversarial modification. It is both possible and practical to create an adversarial input to fool the target DNN model and its coupled interpretation model simultaneously. Fig. 1(c) shows dual adversarial examples resulting from targeting the DNN model and the coupled interpreter. Dual adversarial examples generate interpretation maps indistinguishable from their benign interpretation maps.

Manuscript received 5 April 2023; revised 20 November 2023; accepted 3 December 2023. Date of publication 11 December 2023; date of current version 11 July 2024. An earlier version of this work has appeared in CSoNET 2021 [DOI: 10.1007/978-3-030-91434-9_9]. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by Korea government (MSIT) under Grant 2021R1A2C1011198, in part by (Institute for Information & communications Technology Planning & Evaluation) (IITP) grant funded by Korea government (MSIT) under the ICT Creative Consilience Program under Grant IITP-2021-2020-0-01821, and in part by AI Platform to Fully Adapt and Reflect Privacy-Policy Changes under Grant 2022-0-00688. *(Corresponding author: Tamer Abuhmed.)*

Eldor Abdukhamidov and Tamer Abuhmed are with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 03063, South Korea (e-mail: abdukhamidov@skku.edu; tamer@skku.edu).

Simon S. Woo is with the Department of Artificial Intelligence and Department of Applied Data Science, Sungkyunkwan University, Suwon 03063, South Korea (e-mail: swoo@g.skku.edu).

Mohammed Abuhamad and Eric Chan-Tin are with the Department of Computer Science, Loyola University, Chicago, IL 60660 USA (e-mail: mabuhamad@luc.edu; chantin@cs.luc.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TDSC.2023.3341090, provided by the authors.

Digital Object Identifier 10.1109/TDSC.2023.3341090

In this work, we proposed optimized adversarial attacks that target IDLSes, i.e., can fool the prediction model and the coupled interpreter simultaneously. We call these attacks AdvEdge and AdvEdge$^+$. In particular, the attacks exploit the image's edge information to enable perturbation to be attached to the edges of the image areas spotlighted by the interpretation model's attribution map. This allows for a far more stealthy adversarial attack since the crafted adversarial inputs are difficult to identify, even with the interpretation and human involvement. Furthermore, the proposed attacks generate effective adversarial examples with minimal perturbation.

Our Contribution: This work contributes to the security of IDLes. In particular, our contributions are as follows:

- We propose two novel interpretation-guided attacks against IDLSes, namely, AdvEdge and AdvEdge$^+$, utilizing edge information to enhance the stealthiness of attacks. We evaluate attacks against various DNNs and interpreters (e.g., Grad, CAM, RTS, and MASK).
- We implement our attacks on three frameworks (i.e., PGD, C&W, and StAdv) and evaluate the performance using ImageNet, CIFAR-100, and CIFAR-10 datasets.
- We conduct an analysis of the transferability of adversarial samples across different interpreters.
- We demonstrate the attack persistence against common defenses and present a novel defense method that utilizes multiple interpreters to identify adversarial samples.

Organization: The rest of the paper is organized as follows: Section II highlights the literature; Section III presents fundamental concepts; Section IV provides a description of the proposed attacks; Section V shows the experiments and evaluation; Section VI presents the discussion; and Section VII concludes the paper.

II. Related Work

Interpretability: Interpretability for models can be derived using various methods: back-propagation [8], intermediate representations [9], [10], input perturbation [7], and meta-models [6]. Interpretability can contribute to system security in the decision-making process involving an expert who can inspect/verify the attribution maps contributing to certain outputs. The interpretation of adversarial inputs is expected to differ remarkably from the interpretation of their benign inputs. There have been many works on interpretability to debug DNNs [11] and to identify adversarial inputs [12], [13].

A study [14] shows the feasibility of manipulating IDLSes. It offers a novel attacking method to fool DNN models and their corresponding interpretation models simultaneously, demonstrating that improved interpretability guarantees only limited security and protection. In [15], the authors investigate the privacy risks associated with model explanations to infer sensitive attributes of an input. The study highlights that model explanations represent a notable attack vector, posing a tangible risk to data privacy. In a recent study [16], a backdoor attack was proposed that modifies DNN saliency maps, posing a security threat to interpretation methods. The attack depends on a trigger pattern and cannot be detected by current defenses. In this paper, we introduce two optimized versions of the ADV^2 [14] attack targeting IDLSes.

Transferability: Transferability is an interesting property of adversarial attacks [17]: e.g., adversarial inputs generated for a specific DNN can be effective against other DNNs. This property is used in black-box scenarios [18]: e.g., generating adversarial inputs based on known white-box DNN and applying them against the target black-box DNN [19], [20]. In a research study [21], a technique that perturbs the hidden layers of a model is proposed to improve the transferability of adversarial inputs. Another work [22] applies differentiable transformations to input samples for enhanced transferability. In this work, we investigate the transferability of adversarial inputs across different interpreters.

III. Fundamental Concepts

This section introduces common concepts and terms used in this study. We note that even though this work focuses on IDLSes of classification tasks, such as image classification, the attacks can be generalized to other modeling tasks.

Let a classifier $f(x) = y \in Y$ (i.e., DNN classifier f) that assigns a sample input (x) to a category (y) from a collection of predetermined categories (Y). Let an interpreter $g(x; f) = m$ (i.e., interpretation model g) produce an interpretation map (m) that highlights the feature importance in the sample (x) based on the prediction of the model (f). The value of the i-th component in m (i.e., $m[i]$) represents the importance of the i-th component in the sample x (i.e., $x[i]$)).

From this perspective, we consider that model interpretation can be obtained by two basic methods: ❶ *Post-hoc interpretation*: This can be accomplished by adjusting the complexity of DNNs or by employing post-training techniques. This strategy requires the development of a new model to provide interpretations of the present model [4], [5], [6], [7], [23]. ❷ *Intrinsic interpretation*: This type of interpretability can be produced by developing self-interpretable DNNs, which intrinsically embed the interpretability feature into their structures [4], [5]. In this study, we focus on the first method of interpretability, in which an interpreter (g) obtains an attribution map m of how a DNN f identifies the sample x.

In a typical adversarial setting, DNN models are vulnerable to adversarial examples [17], [24]. More specifically, an adversarial input ($\hat{x}$) can be generated by manipulating the benign input (x) using one of the well-known attacks (i.e., PGD [25], BIM [26], C&W [27], StAdv [28], etc.) to make the model misclssify $\hat{x}$ to an output $f(\hat{x}) = y_t \neq f(x)$. The manipulations, e.g., also known as adversarial perturbations, are usually bounded by a norm ball $\mathcal{B}_\varepsilon(x) = \{\|\hat{x} - x\|_\infty \leq \varepsilon\}$, where ε is the threshold, to ensure its success and evasiveness.

In this work, we adopt the following three well-known attack frameworks in our paper:

❶ *PGD:* A first-order adversarial attack applies a sequence of project gradient descent on the loss function

$$\hat{x}^{(i+1)} = \prod_{\mathcal{B}_\varepsilon(x)} \left(\hat{x}^{(i)} - \alpha . \, sign(\nabla_{\hat{x}} \ell_{prd}(f(\hat{x}^{(i)}), y_t)) \right). \quad (1)$$

Here, $\prod$ is a projection operator, $\mathcal{B}_\varepsilon$ is a norm ball restrained by a pre-fixed ε, α is a learning rate, x is the benign sample, $\hat{x}^{(i)}$ is the $\hat{x}$ at the iteration i, and ℓ_{prd} is a loss function that calculates the difference between $f(\hat{x})$ and y_t.

❷ *C&W.* C&W attack framework considers the process of generating adversarial inputs as an optimization problem of L_p-norm regarding the distance metric δ with respect to the given input x, which can be described as

$$||\boldsymbol{\delta}||_p = \left(\sum_{i=1}^{n} |\delta_i|^p\right)^{1/p} \quad ; \quad \delta_i = \hat{x}_i - x_i$$

$$\text{minimize } ||\boldsymbol{\delta}||_p + c \cdot \ell(x+\delta) \quad \text{s.t.} \quad x+\delta \in [0,1]^n, \tag{2}$$

where δ is the perturbation added to x, ℓ is a loss function to solve the optimization problem using gradient descent, and c is a constant. We utilize L_2-norm based C&W in this work.

❸ *StAdv:* The attack is based on the spatial transformation of pixels to improve the perceptual quality of adversarial samples. The per-pixel flow is expressed as the flow vector: $r_i := (\Delta u^{(i)}, \Delta v^{(i)})$, specifically, $\Delta u^{(i)} = u_x^{(i)} - u_{\hat{x}}^{(i)}$ and $\Delta v^{(i)} = v_x^{(i)} - v_{\hat{x}}^{(i)}$, where $(u_x^{(i)}, v_x^{(i)})$ and $(u_{\hat{x}}^{(i)}, v_{\hat{x}}^{(i)})$ represent the 2D coordinates of the ith pixel of x and $\hat{x}$. Obtaining the pixel values of the adversarial input is described as

$$\hat{x}^{(i)} = \sum_{q \in \mathcal{N}([u_x^{(i)}, v_x^{(i)}])} x^{(q)} (1 - |u_x^{(i)} - u_x^{(q)}|)(1 - |v_x^{(i)} - v_x^{(q)}|), \tag{3}$$

where $\mathcal{N}([u_x^{(i)}, v_x^{(i)}])$ represents the indices of the four-pixel neighbors (top/bottom-left/right) of $(u_x^{(i)}, v_x^{(i)})$. The following two loss functions are used to formulate the search of adversarial deformations as an optimization problem, and they are described as follows:

$$\ell_{prd}(x,r) := \max\left(\max_{i \neq t}(f_i(x+r)) - f_t(x+r), \kappa\right),$$

$$\ell_{flow}(r) := \sum_p \sum_{q \in \mathcal{N}(p)} \left(||\Delta u^{(p)} - \Delta u^{(q)}||_2 + ||\Delta v^{(p)} - \Delta v^{(q)}||_2\right)^{\frac{1}{2}},$$

where κ is the misclassification confidence, p is an arbitrary pixel and $q \in \mathcal{N}(p)$ represents the neighbors of the pixel p.

The optimization problem is balanced with a factor of λ between the two losses to encourage the adversarial input to be classified while preserving high perceptual quality, as follows:

$$\min_r \left\{\ell_{prd}(x,r) + \lambda \ell_{flow}(r)\right\}. \tag{4}$$

IV. AdvEdge Attack

The section provides the details of implementing the proposed attacks against four different types of interpreters.

A. Threat Model

Adversarial Objective: The main aim of AdvEdge and AdvEdge$^+$ is to fool the target IDLS (i.e., both classifier and interpreter) by introducing stealthy adversarial perturbations using edge information of input images (i.e., adding noise to edges). The goal is to maximize the rate of misclassification while ensuring that adversarial perturbations remain imperceptible and do not alter the attribution of the original image. In general, there are two reasons for applying perturbations to the edges of images: ❶ *Edge Sensitivity:* DNNs are usually sensitive to edge information, and altering these edges can contribute to fooling DNNs. ❷ *Stealthiness:* Generally, changes in edges (i.e., significant transitions of colors or intensities) are less noticeable to humans than changes in smoother "*uniform*" regions (i.e., large areas of the same color or texture), making the attack less perceptible. Moreover, limiting perturbations to edges could minimize the rate of perturbations that achieve the adversarial objective (see Fig. 5 and Appendix V-D), available online.

Adversarial Capabilities: We explore different adversarial capabilities for AdvEdge and AdvEdge$^+$ attacks as follows:

❶ *White-box scenario* in which the adversary has full knowledge and access to the victim model and its interpreter. This provides the attacker with the highest level of capability to generate adversarial perturbations for maximum effect.

❷ *Gray-box scenario* in which the adversary has full access to the victim classifier, but no access or knowledge about the interpreter. The attacker uses the transferability property of interpretations to mislead the target interpreter.

B. Attack Definition

The adversarial goal is achieved by creating an adversarial sample $\hat{x}$ that meets the following conditions: 1) A DNN f misclassifies the adversarial sample $\hat{x}$ to y_t: $f(\hat{x}) = y_t$ so that $f(\hat{x}) = y_t \neq f(x)$. 2) The adversarial sample $\hat{x}$ causes the interpretation model g to generate the target interpretation map m_t: $g(\hat{x}; f) \approx g(x; f)$. 3) The adversarial and benign samples are indistinguishable, i.e., constraining the noise to the edges within the image.

The proposed approach aims to discover a minimal perturbation so that adversarial examples satisfy these conditions. The optimization framework to characterize the attack is

$$\min_{\hat{x}} : \Delta(\hat{x}, x) \text{ s.t. } \begin{cases} f(\hat{x}) = y_t \\ g(\hat{x}; f) = m_t \end{cases}.$$

This can be reformulated to be more appropriate for optimization as follows:

$$\min_{\hat{x}} : \ell_{prd}(f(\hat{x}), y_t) + \lambda . \ell_{int}(g(\hat{x}; f), m_t) \text{ s.t. } \Delta(\hat{x}, x) \leq \varepsilon, \tag{5}$$

where ℓ_{prd} and ℓ_{int} denote the classification loss as in (1) and the interpretation loss, respectively. ℓ_{int} quantifies the dissimilarity between adversarial and benign attribution maps, i.e., $g(\hat{x}; f)$ and $m_t = g(x; f)$, respectively. The hyperparameter λ is used to balance the two components (i.e., ℓ_{prd} and ℓ_{int}). We build (5) based on the PGD framework, and it can be applied to the other attack frameworks with few modifications (i.e., considering (2) and (4)) as follows:

$$\min_{\hat{x}} : \ ||\boldsymbol{\delta}||_p + c\, \ell_{prd}(f(\hat{x}), y_t) + \lambda . \ell_{int}(g(\hat{x}; f), m_t) \tag{6}$$

$$\min_{\hat{x}} : \ell_{prd}(f(\hat{x}), y_t) + \lambda . \ell_{int}(g(\hat{x}; f), m_t) + \lambda . \ell_{flow}(r). \tag{7}$$

The terms are defined as

$$\ell_{prd}(f(\hat{x}), y_t) = -\log(f_{y_t}(\hat{x})),$$
$$\Delta(\hat{x}, x) = \| \hat{x} - x \|_\infty, \text{ and}$$
$$\ell_{int}(g(\hat{x}; f), m_t) = \|g(\hat{x}; f) - m_t\|_2^2.$$

In PGD, the perturbation in (1) is controlled as

$$\hat{x}^{(i+1)} = \prod_{\mathcal{B}_\varepsilon(x)} \left(\hat{x}^{(i)} - N_w \, \alpha . sign(\nabla_{\hat{x}} \ell_{adv}(\hat{x}^{(i)})) \right). \tag{8}$$

For C&W and StAdv attack frameworks, we control the perturbation in (2) and (4) as follows:

$$\delta = N_w \, (\hat{x} - x), \tag{9}$$
$$r := N_w \, (\Delta u, \Delta v), \tag{10}$$

where, N_w is the perturbation method that determines both the amount and position of noise injected to the sample considering the edge weights w of the benign sample. The entire loss equation (i.e., (5)) is represented as ℓ_{adv}.

AdvEdge: In (8), we use the N_w term to optimize the position and amount of applied perturbation. In AdvEdge, we limit the added perturbation to the edges of the sample that overlap with the interpretation map provided by the interpreter. In other words, we learn the critical parts of a given input sample based on the overall loss (i.e., DNN loss and interpretation model loss) and then inject noise into the edges of those areas. We define the edges based on the edge weights acquired by applying the Sobel filter [29]) to the sample image.

A pixel-wise edge weight matrix is defined as $\mathcal{E} : e \rightarrow \mathbb{R}^{h \times w}$ for an image of height h and width w using a typical edge extraction method (e.g., Sobel filter in our experiments). These weights are applied to the sign of the gradient update (PGD framework) as follows: $\mathcal{E}(x) \otimes \alpha . sign(\nabla_{\hat{x}} \ell_{adv}(\hat{x}^{(i)}))$, where the operator $\otimes$ represents the Hadamard product. By doing so, the attack amplifies the noise on the edges while reducing or constricting its presence elsewhere in the image.

Considering a straightforward application of the edge weight matrix, i.e., $N_w = \mathcal{E}(x)$, we can rewrite (8), (9), and (10) as: $\hat{x}^{(i+1)} = \prod_{\mathcal{B}_\varepsilon(x)} (\hat{x}^{(i)} - \mathcal{E}(x) \, \alpha . sign(\nabla_{\hat{x}} \ell_{adv}(\hat{x}^{(i)})))$, $\delta = \mathcal{E}(x) \times \, (\hat{x} - x)$, and $r := \mathcal{E}(x) \, (\Delta u, \Delta v)$, respectively.

AdvEdge$^+$: This technique, like AdvEdge, leverages the edge weights to optimize the perturbation. Instead of adjusting the noise based on the edge weights in the attack, we apply the perturbations only on defined edges that pass a certain threshold. This is performed by a binarization operation (e.g., $bin(\mathcal{E}_\varphi(x) \rightarrow e_i := \{0 \text{ if } e_i \leq \varphi; \, 1 \text{ otherwise } \forall i\})$) to obtain a binary edge matrix $\mathcal{E}_\varphi : e \rightarrow [0, 1]^{h \times w}$, where h and w are the height and width of an input, respectively. The hyperparameter φ sets the threshold for binarizing edge weights and can be adjusted according to the attack objective. The rest of the implementation is similar to AdvEdge, e.g., the PGD attack can be formulated as: $\mathcal{E}_\varphi(x) \otimes \alpha . sign(\nabla_{\hat{x}} \ell_{adv}(\hat{x}^{(i)}))$. This enables perturbations to be applied only on the edges. Given the constraint on the locality and amount of perturbation, the threshold φ is adjusted to 0.1 to improve the effectiveness. Source code for the implementation can be found in the following link: *https://github.com/InfoLab-SKKU/AdvEdge-Attack*.

The implementation against representatives of four categories of interpreters is discussed in the following subsections.

C. Backpropagation-Guided Interpretation

This category of interpretation models computes the gradient of a DNN model's prediction with respect to the provided sample input. This highlights the significance of each feature in the input sample. Larger values suggest higher importance of the features to the model prediction. In this paper, we discuss gradient saliency (Grad) [23] as an example of backpropagation-guided interpretation. The Grad interpreter estimates an attribution map m given the model f and input x with a class y as: $m = |\frac{\partial f_y(x)}{\partial x}|$.

Looking for the optimal adversarial sample $\hat{x}$ for Grad-based IDLSes through a sequence of gradient descent updates (as in applying (8)) is ineffective since DNN models with ReLU activation functions cause the Hessian matrix computation result to be all-zero. The challenge can be resolved by computing the smoothed value of the gradient of ReLU ($r(z)$) using a function $h(z)$ with the following formula:

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

Here, τ is a constant parameter, and $h(z)$ approximates the values of $r(z)$ and its gradients are non-zero. Another alternative is to use the sigmoid function ($\sigma(z) = \frac{1}{1+e^{-z}}$). We note that this method can be applied to other backpropagation-guided interpreters (e.g., LRP [30], SmoothGrad [31]) as they are based on gradient-centric formulations [32].

D. Representation-Guided Interpretation

This type of interpreter extracts feature maps from intermediate layers of DNN to provide attribution maps. Representative of representation-guided interpretation in this study is **C**lass **A**ctivation **M**ap (CAM) [10]. CAM performs global average pooling on the convolutional feature maps and utilizes the output as features for a fully-connected layer for the model prediction. The significance of the sample input areas can be determined by projecting the output layer weights back onto the convolutional feature maps.

Specifically, let $a_i(k, l)$ be the activation of channel i in the last convolutional layer at (k, l) spatial location, and $\sum_{k,l} a_i(k, l)$ as the result of global average pooling. The activation map m_c for an input x and class c is given as $m_c(x, y) = \sum_i w_{i,c} a_i(k, l)$, where c is cth output of the linear layer and $w_{i,c}$ is the weight corresponding to class c for ith channel. We construct m by extracting and concatenating interpretation maps from f up to the final convolutional layer and a fully connected layer, similar to the work of [14]. We exploit the interpretation model by utilizing gradient descent updates to find the optimal $\hat{x}$ as in (8). The attack can extend to various interpreters of this category (e.g., Grad-CAM [9]).

E. Model-Guided Interpretation

By masking significant areas in any input, a model-guided interpretation model is trained to predict the attribution map directly in a single forward pass. We consider **R**eal-**T**ime Image **S**aliency (RTS) [6] for this category. RTS generates the attribution map m for any given class c and input x by resolving the following problem:

$$\begin{aligned}\min_{m}: \quad & \lambda_1 r_{tv}(m) + \lambda_2 r_{av}(m) \\ & - \log(f_c(\Phi(x;m))) \\ & + \lambda_3 f_c(\Phi(x;1-m))^{\lambda_4} \ \text{ s.t. } 0 \leq m \leq 1. \end{aligned} \tag{11}$$

Here, the regularizers $\{\lambda_i\}_{i=1}^{4}$ balance the involved factors, and $r_{av}(m)$ represents the average value of m. The $r_{tv}(m)$ expresses the variation of m that enforces mask smoothness, and it is defined simply as: $r_{tv}(m) = \sum_{i,j}(m_{i,j} - m_{i,j+1})^2 + \sum_{i,j}(m_{i,j} - m_{i+1,j})^2$. Furthermore, Φ is the operator that uses m as a mask to blend x with random color noise and Gaussian blur, and it is defined as: $\Phi(x,m) = x \otimes m + r \otimes (1-m)$, where $\otimes$ denotes the Hadamard product, and r is an alternative image. For applying Φ, r can be chosen to be a highly blurred version of in the input x. In general, (11) finds the important parts of x in terms of which f predicts $f(x)$ with high confidence.

During the inference, computing (11) is expensive. Therefore, RTS trains a DNN to predict the attribution map for any input so that RTS does not have to access the DNN f after training. This can be achieved by composing a ResNet [2], a pre-trained model as an encoder to extract feature maps of the given inputs and a U-Net [33] as the masking model which is trained to optimize the framework in (11). For this work, we consider the composition of the encoder and masking model as the interpreter g.

Attacking RTS by employing (8), (9) or (10) directly, has been found to be inefficient for finding the optimal adversarial samples [14]. The main reason for this is that the interpretation model uses both the masking model and the encoder $(enc(\cdot))$. To address this issue, we utilize an additional loss term $\ell_{enc}(enc(\hat{x}), enc(y_t))$ to (5), (6) and (7) to compute the difference between the encoder's outcome with the adversarial sample input $\hat{x}$ and the target class y_t. Then, we perform a series of gradient descent updates to choose the optimal adversarial sample input $\hat{x}$.

F. Perturbation-Guided Interpretation

This type of interpretation model investigates interpretation maps by introducing a minimal amount of noise to the input and evaluating the changes in the model prediction. We consider MASK [7] to be the class representative.

MASK identifies the attribution map for the given input x by affecting the maximally informative input regions. Specifically, the model $f(x)$ generates a vector of scores for distinct hypotheses about input x (e.g., as a softmax probability layer in a DNN), then MASK explores the smallest mask m to make the model performance drop significantly: i.e., $f_y(\Phi(x;m)) \leqslant f_y(x)$. We note that the mask m in this scenario is binary, where $m[i] = 0$ or 1 to indicate whether the i-th feature is replaced with Gaussian noise. The optimal mask can be obtained by solving the following problem:

$$\min_{m}: f_y(\Phi(x;m)) + \lambda. \|1-m\|_1 \ \ s.t.\ 0 \leq m \leq 1, \tag{12}$$

where λ encourages most of the mask to be sparse. By solving the (12), we find the most informative and necessary regions of the input x with reference to the model prediction $f(x)$.

We cannot directly advance (5), (6) and (7) with (8), (9) and (10), respectively, since the interpretation model g is built as an optimization method.

A bi-level optimization framework [14] can be used to solve this problem. The loss function is rewritten as follows: $\ell_{adv}(x,m) \triangleq \ell_{prd}(f(x), y_t) + \lambda. \ell_{int}(m, m_t)$ by including benign attribution map m as a new variable. Let $\ell_{map}(m;x)$ denote the object function of (12) and $m_*(x) = \text{argmin}_m : \ell_{map}(m;x)$ be the attribution map generated by MASK for the input x. Then, we have the following framework:

$$\min_{x}: \ell_{adv}(x, m_*(x))\ s.t.\ m_*(x) = \underset{m}{\text{argmin}}\ \ell_{map}(m;x).$$

For every update on the input x, solving this bi-level optimization is expensive. As a solution to the problem, [14] proposed an approximate iterative procedure to optimize x and the mask m by alternating between ℓ_{adv} and ℓ_{map}. Briefly, given $x^{(i-1)}$ at the ith iteration, the attribution map $m^{(i)}$ is computed by updating $m^{(i-1)}$ in $\ell_{map}(m^{(i-1)}; x^{(i-1)})$, then $m^{(i)}$ is fixed and $x^{(i)}$ is attained by reducing ℓ_{adv} after a gradient descent step with respect to $m^{(i)}$. To update $x^{(i)}$, the objective function is formulated as follows:

$$\ell_{adv}\left(x^{(i-1)}, m^{(i)} - \alpha. \nabla_m. \ell_{map}(m^{(i)}; x^{(i-1)})\right), \tag{13}$$

where, α is the learning rate.

V. Experiments and Evaluation

A. Experimental Settings

AdvEdge and AdvEdge$^+$ are built based on the PGD, C&W, and StAdv attack frameworks. We use the values $\alpha = 1/255$ that is mentioned in (13), and $\varepsilon = 0.031$ as the perturbation threshold from the prior work [14]. ℓ_∞ is used to calculate the perturbation proportion. To improve the efficiency of the attack, we use a method that adds noise to the edges of the images with a fixed number of iterations (#iterations $= 300$).

Optimization Steps: Zero gradients of the prediction loss hinder finding the desired result with correct interpretation (Grad). To address this issue, a label smoothing strategy using cross-entropy is used. The approach samples the prediction loss using a uniform distribution $\mathbb{U}(1-\rho, 1)$ and the value of ρ is substantially decreased during the attack process. Considering $y_c = \frac{1-y_t}{|Y|-1}$, we derive $\ell_{prd}(f(x), y_t) = -\sum_{c \in Y} y_c \log f_c(x)$.

In solving (13), multiple steps of gradient descent are applied to update m and calculate $m_*(x)$ for faster convergence. Additionally, the average gradient is used to update m and support stable optimization. More precisely, at the ith iteration with multistep gradient descent, $\{m_j^{(i)}\}$ is the sequence of maps. To calculate the gradient to update m, the interpretation loss

$\sum_j \|m_j^{(i)} - m_t\|_2^2$ is used. To improve convergence, the learning rate is also dynamically changed by applying the Adam optimizer to calculate the optimal learning rate at each iteration. For stable learning, x is updated in two steps: ❶ updating x based on prediction loss ℓ_{prd} and interpretation loss ℓ_{int}; ❷, the confidence score is checked if it is still above a specific threshold (i.e., 0.9 for our experiment) after perturbation by searching the largest step size (i.e., maximum 0.08 in our case) in the adversarial search space.

Since MASK is considered an optimization procedure, unlike other categories of interpreters, we adopt additional steps to find the optimal adversarial sample. Update of the estimate of the attribution map in terms of ℓ_{map} results in a significant deviation from the original MASK map as the number of steps increases in the update process. To maintain the effectiveness of the attack, the generated map is replaced with the map $\hat{m}^{(i)} = g(\hat{x}^{(i)}; f)$ which is the MASK output in terms of the latest adversarial input at a specific iteration point (e.g., every 50 iterations). Simultaneously, when the estimated map is replaced by the algorithm, we reset Adam's step parameter to have a correct internal state and decrease the negative impacts on the attack performance.

Dataset: For our experiment, we use ImageNetV2 Top-Images [34] dataset, CIFAR-10 and CIFAR-100 datasets [35]. ImageNetV2 is a new test set collected based on the ImageNet benchmark and was mainly published for inference accuracy evaluation. The dataset contains 10,000 images based on 1,000 different classes that are similar to the classes in the original ImageNet dataset. All images are cropped to 224 $\times$ 224 pixels, and the pixels are normalized between $[0, 1]$. As the test set, we use 1,000 images (a total of 3,000 images) that are selected randomly from each category and classified correctly by the classifier f. The CIFAR-10 dataset comprises 60,000 color images, each measuring 32x32, and is organized into 10 categories, with 6,000 images per category. Out of these, 50,000 are designated for training and 10,000 for testing. CIFAR-100 is similar to CIFAR-10, but features 100 categories, each containing 600 images. For each category, 500 images are used for training, while the remaining 100 are for testing.

Prediction Models: The study employs four state-of-the-art pre-trained DNN models, *ResNet-50* [2], *DenseNet-169* [36], *VGG-16* [37], and *Inception-V3* [38] which demonstrate 22.85%, 22.08%, 24.40%, and 21.20% top-1 error rates on the ImageNet dataset, respectively. These DNN models differ in capacity (50, 169, 41, and 48 layers, respectively) and network architecture (residual blocks, dense blocks, *etc.*). The selected models are popular and extensively employed in a wide range of applications. Their distinct architectures and capabilities allow for an evaluation of the suggested attack techniques across various models, offering a thorough examination of the attacks' effectiveness and applicability.

Interpretation Models: We explored the attacks against the Grad [23], CAM [10], RTS [6], and MASK [7] interpreters as representative of back-propagation-guided, representation-guided, model-guided, and perturbation-guided interpretation models, respectively. In our experiment, we employed the interpreters' original open-source implementation.

Comparison of Attack Approaches: We compare AdvEdge and AdvEdge$^+$ with ADV^2 attack against IDLes. For a fair comparison, we use the same hyperparameters (e.g., *learning rate*, *number of iterations*, and other experimental settings for all approaches. We note that comparing our attacks to other recent attacks against IDLes, e.g., [39], [40], can be infeasible. For instance, attack [40] operates on tabular datasets while our attack is implemented specifically for image-based inputs focusing on edges for perturbation injection. Another relevant work [39] focuses on manipulating interpretation and training a classifier while maintaining the accuracy of the output. Since our attack aims to fool classifiers and interpreters without modifying them, a direct comparison can be impractical.

Evaluation Metrics: We used several evaluation metrics to assess the attack success against the used DNN classifiers and interpretation models. To begin, we use the following metrics to evaluate the attacks based on misleading DNN models.

- *Attack success rate:* We calculate the proportion of successfully misclassified test inputs to the total number of test samples, which is calculated as $\frac{\#\text{successful trials}}{\#\text{total trials}}$.
- *Misclassification confidence:* We check the prediction confidence of the output, which is the probability given to the class by a DNN model.

Moreover, we assess attacks based on their ability to deceive the interpretation model. This is achieved by evaluating the interpretation maps of adversarial samples. This task is challenging since there is a lack of standard metrics to evaluate the interpretation maps provided by the interpreters. Therefore, we use the following metrics:

- *Qualitative comparison*: We use this measurement to check whether the results of our approach are perceptually indistinguishable. We qualitatively compare the interpretation maps of benign and adversarial inputs.
- $\mathcal{L}_p$ *Measure*: We observe the difference between benign and adversarial attribution maps using the $\mathcal{L}_1$ distance. To this end, the values of the attribution maps are standardized in the range of $[0, 1]$.
- *IoU Test* (**I**ntersection-**o**ver-**U**nion): This is another quantitative metric for determining the similarity of the attribution maps. This metric is commonly used to compare the prediction outcome with the ground truth: $\text{IoU}(m) = |O(m) \bigcap O(m_\circ)| / |O(m) \bigcup O(m_\circ)|$, where m is adversarial attribution map and $m_\circ$ is the benign attribution map. $O(m)$ indicates the set of non-zero dimensions in m.

The following metrics are also used to evaluate attacks.

- *Structural Similarity (SSIM)*: The mean structural similarity index [41] between the benign and adversarial samples is used to calculate the added noise. SSIM measures the image quality based on its distortion-free reference image. To calculate the non-similarity rate (also known as the distance or noise rate), we subtract the SSIM value from one (noise rate $= 1 -$ SSIM).
- *Average Time*: We calculate the time that is taken to generate adversarial inputs by the attacks.

B. Attack Effectiveness Against DNNs

In terms of fooling the target DNN models, we first evaluate AdvEdge and AdvEdge$^+$ along with comparing the results with the previous work (ADV^2 [14]). Since ADV^2 applies PGD, we adapt the attack to C&W and StAdv frameworks for comparison purposes. The attack success rate of our attacks is 100% when applying PDG, C&W, and StAdv against all models and interpreters in CIFAR-10 and CIFAR-100. When using the ImageNet dataset, the attack success rates are 100% for all attacks when applying PGD, around 98.4%, 98.4%, 98.3% for ADV^2, AdvEdge, AdvEdge$^+$ respectively when applying C&W, and 97.9%, 97.7%, 97.5% for ADV^2, AdvEdge, AdvEdge$^+$, respectively when applying StAdv framework.

In terms of misclassification confidence (MC), our attacks outperform existing attacks. In ImageNet, the average MCs are 0.684 and 0.680 for our attacks compared to ADV^2 with 0.671 when applying PGD. When applying C&W, the MCs are 0.689, 0.700, and 0.696 for ADV^2, AdvEdge and AdvEdge$^+$, respectively. In the case of applying StdAdv, the MC for ADV^2 is 0.752 compared to our attacks with 0.757 and 0.753. In CIFAR-100 and CIFAR-10, the average MCs are higher than the MC of ImageNet. In general, in CIFAR-100, the average MC for ADV^2 is 0.930 in different frameworks, while our AdvEdge and AdvEdge$^+$ attacks have 0.934 and 0.935.

In CIFAR-10, the MCs are 0.928, 0.933, and 0.933 for ADV^2, AdvEdge, and AdvEdge$^+$, respectively. From the results, we found that even though the noise rate in our attacks is small, our attack methods still achieved equal or higher results compared to the existing attack. This highlights the efficacy of our attack methods in generating adversarial examples with minimal perturbations. Detailed results in terms of *attack success rate* and *misclassification confidence* are shown in Table VII and Table VIII in Appendix A, available online.

C. Attack Effectiveness Against Interpreters

We assess the capability of AdvEdge and AdvEdge$^+$ attacks to create interpretation maps that are comparable to benign interpretation maps. Figs. 2 and 9 visualize several examples for ImageNet and CIFAR datasets, respectively, when the ResNet, VGG, and Inception models are used together with all interpreters. In the figures, our attacks are compared with the existing attack. We begin with a qualitative evaluation to observe the similarity of interpretations produced by adversarial samples (i.e., generated by AdvEdge and AdvEdge$^+$) and the corresponding benign samples. By making a qualitative comparison of the interpretations of benign and adversarial samples, AdvEdge and AdvEdge$^+$ generated interpretations that are visually indistinguishable from their corresponding benign sample inputs. Compared to ADV^2, both proposed approaches produced interpretation maps that are similar to benign inputs. Fig. 2 shows some examples of observed attribution maps obtained using Grad, CAM, RTS, and MASK. The examples show adversarial interpretations and corresponding benign ones. As shown in the figure, the adversarial and benign attribution maps are highly similar.

Moreover, we employ $\mathcal{L}_p$ to quantify the similarity of generated interpretation maps. Fig. 3 summarizes the results of

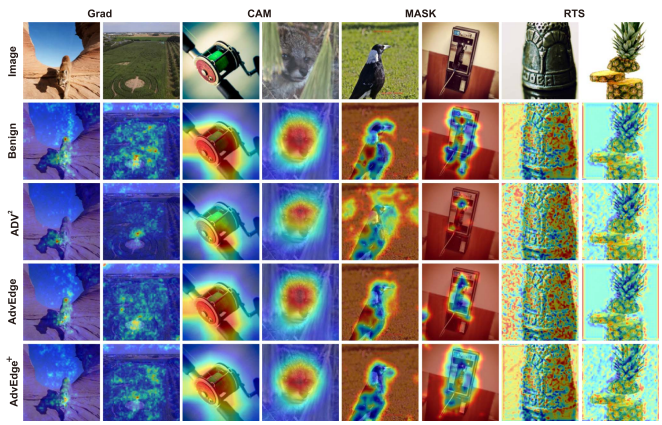

Fig. 2. Examples of benign and adversarial attribution maps generated by Grad, CAM, MASK, and RTS interpreters on the ResNet model. Adversarial attribution maps are based on ADV^2, AdvEdge, and AdvEdge$^+$ attacks.

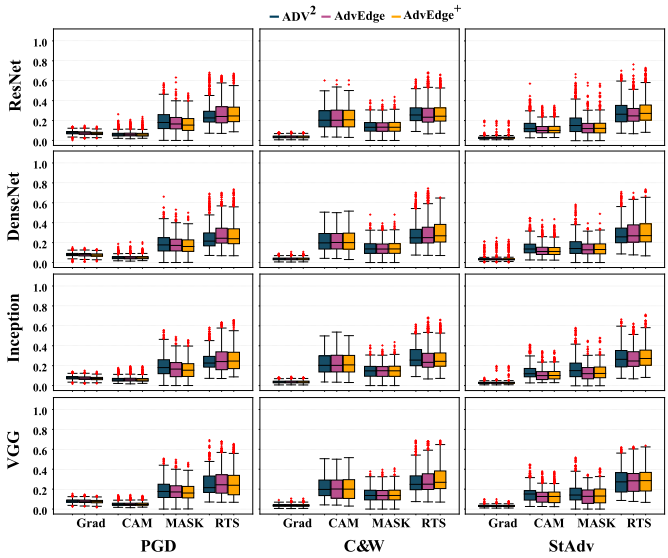

Fig. 3. $\mathcal{L}_1$ adversarial-to-benign distance of attribution maps by different interpreters on various DNNs using ADV^2, AdvEdge, and AdvEdge$^+$ for PGD, C&W, and stAdv using ImageNet.

$\mathcal{L}_1$ measurement using the ImageNet dataset (Note: the results for CIFAR-10 and CIFAR-100 datasets are in Figs. 10 and 11, respectively). Compared to ADV^2, AdvEdge and AdvEdge$^+$ produce adversarial samples with interpretation maps that are similar to those obtained for benign samples. The results are consistent among the interpretation models with all DNNs. We observe that the efficiency of our attack (against interpretation models) differs from the interpretation models. Furthermore, as reflected in the Fig. 3, there are several cases that the attacks could not achieve low $\mathcal{L}_1$ distance (shown as red marks in the figure), which are known as outliers. We analyze such cases in Section VI.

The IoU score is another quantitative metric for comparing the similarity of interpretation maps. We binarized the values of the interpretation maps to compute the IoU as they are originally real

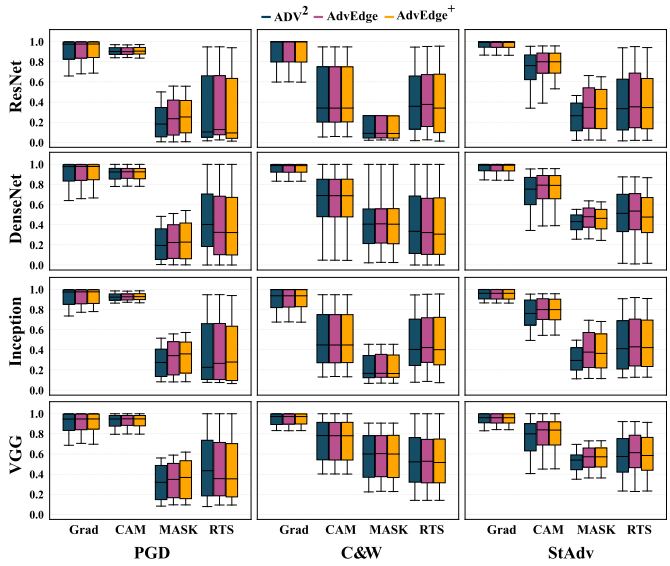

Fig. 4. IoU of adversarial and benign attribution maps by different interpreters on various DNNs using ADV^2, AdvEdge, and AdvEdge$^+$ for PGD, C&W, and stAdv using ImageNet.

values. Fig. 4 displays the IoU scores of attribution maps generated by ADV^2, AdvEdge, and AdvEdge$^+$ adopting PGD, C&W, and StAdv with regard to four interpreter models on ResNet, DenseNet, VGG and Inception using the ImageNet dataset. Both CIFAR-10 and CIFAR-100 share similar characteristics, such as image size, resolution, and complexity; therefore, they provide similar results. The results for CIFAR-10 and CIFAR-100 datasets are in Figs. 12 and 13. As shown in the figures, AdvEdge and AdvEdge$^+$ performed better than ADV^2 using different datasets and models. We observe that AdvEdge and AdvEdge$^+$ achieved significantly better results in terms of the Grad, CAM, and MASK interpreters. AdvEdge and AdvEdge$^+$ are effective in deceiving different interpreters. Based on qualitative and quantitative measures, the attribution maps of adversarial and benign samples are almost indistinguishable.

D. Adversarial Perturbation Rate

SSIM helps assess the pixel-level perturbation applied by AdvEdge and AdvEdge$^+$ to generate adversarial samples. We measure the amount of noise from images' areas that are not identical to the original images using SSIM. Fig. 5, Figs. 14 and 15 present the amount of perturbation generated by different attacks to achieve success in different frameworks using ImageNet, CIFAR-100, and CIFAR-10 datasets, respectively. As shown, AdvEdge and AdvEdge$^+$ required significantly lower noise than ADV^2 in terms of PGD and C&W. The results become more obvious when the MASK interpreter is used (PGD attack). AdvEdge$^+$ generates the least amount of noise to fool the target DNN model and the MASK interpreter. In terms of StAdv, limiting the perturbation to only the edges of specific regions caused the attack to use more noise to have a successful attack compared to ADV^2. This can be more noticeable when a MASK interpreter is used.

E. Average Time

To compare attacks in terms of time, we calculated the time required to generate adversarial input. As shown in Table I, the CAM and RTS interpreters took less time for the generation process compared to Grad and MASK in general. Among the four interpreters, MASK is the interpreter that takes considerable time to achieve a successful attack. In terms of attack frameworks adopted (PGD, StAdv, and C&W), C&W based on CAM and MASK is faster, while PGD based on Grad generates adversarial in a shorter time period. According to the types of attacks, there is no significant time difference among ADV^2, AdvEdge, and AdvEdge$^+$. In several cases, our attacks took less time to search in adversarial space (Grad). This can be clearly observed in the time taken by AdvEdge and AdvEdge$^+$ using StAdv on the MASK interpreter and ResNet classifier. Due to the small size of the images in CIFAR-100 and CIFAR-10 datasets, the time required for an attack is significantly lower compared to the ImageNet dataset as expected. Generally, CAM and RTS interpreters require less time to deceive compared to Grad and MASK interpreters. Similar patterns can be observed across attack frameworks.

F. Attack Transferability

One interesting property of adversarial inputs is their transferability. Specifically, an effective adversarial input against a DNN can be effective against other DNNs [19], [20], [24]. Similarly, we examine the transferability of our attacks across interpreters. For a given interpreter g, a set of adversarial inputs generated against g is randomly selected (i.e., 100 samples) to compute their attribution maps using the other interpreters g'. Fig. 6 illustrates the attribution maps of adversarial inputs generated against g (using PGD, C&W, StAdv attacks) and transferred to the target g'. We provide the example for the PGD attack, as it adds a considerable amount of noise to the inputs for high attack success, which is highly likely for the attribution maps to deviate from their original attribution maps (see Fig. 5). Additionally, we compare the attribution maps of adversarial inputs with their corresponding original attribution maps. As shown in Fig. 6, the transferability of interpretation is valid for attacks, and it is more obvious in AdvEdge and AdvEdge$^+$ than in ADV^2.

On the quantitative side, Table II shows the $\mathcal{L}_1$ distance between attribution maps of adversarial (ADV^2, AdvEdge and AdvEdge$^+$) and benign samples across different interpreters. Observe that our methods generate better-quality interpretation maps on different interpretation models g compared to the samples produced by ADV^2. AdvEdge and AdvEdge$^+$ maintain high transferability across interpreters for most classes.

G. Attack Effectiveness Against DNNs With Defenses

We conducted experiments to evaluate the effectiveness of the AdvEdge attack against three different defense techniques: Median Smoothing (MS), Bit Squeezing (BS), and Adversarial Training (AT). We chose AdvEdge for this experiment because the previous tables showed that AdvEdge and AdvEdge$^+$ have

TABLE I

AVERAGE TIME (IN SECONDS) TO GENERATE AN ADVERSARIAL INPUT BY ADV^2, ADVEDGE AND ADVEDGE$^+$ ACROSS DIFFERENT INTERPRETERS AND DNNS FOR IMAGENET DATASET USING PGD, C&W, AND STADV

		ImageNet											
		Grad			CAM			MASK			RTS		
		ADV^2	**AdvEdge**	**AdvEdge$^+$**	**ADV^2**	**AdvEdge**	**AdvEdge$^+$**	**ADV^2**	**AdvEdge**	**AdvEdge$^+$**	**ADV^2**	**AdvEdge**	**AdvEdge$^+$**
PGD	**Resnet**	32.68	32.69	**30.02**	15.37	16.05	15.83	188.97	**188.14**	194.56	8.75	**8.45**	**8.39**
	Densenet	33.02	**33.01**	**31.45**	49.42	**49.38**	**49.37**	340.49	343.40	340.53	12.43	**12.37**	16.73
	Inception	32.86	**32.80**	**32.51**	46.25	**46.12**	**46.22**	336.15	338.28	336.42	10.76	**10.75**	**10.63**
	VGG	26.14	**26.01**	**25.73**	12.84	12.86	12.91	150.39	**149.87**	**150.08**	7.47	**7.32**	**7.45**
C&W	**Resnet**	75.95	**75.68**	**75.64**	4.68	**4.68**	4.72	174.18	185.32	**139.37**	8.78	8.96	8.95
	Densenet	173.70	**173.09**	**173.26**	4.46	**4.46**	4.54	201.81	**201.78**	202.91	12.29	**12.23**	12.56
	Inception	140.76	**139.62**	**138.39**	3.52	**3.45**	**3.51**	172.21	**171.59**	**170.97**	8.96	**8.70**	**8.75**
	VGG	58.56	**58.01**	58.60	3.01	**2.57**	3.03	162.78	**161.05**	**162.70**	7.44	**7.37**	**7.38**
StAdv	**Resnet**	80.16	**79.88**	**79.96**	13.13	**13.12**	13.17	987.88	**731.46**	**732.35**	7.48	7.68	7.82
	Densenet	179.17	**179.17**	179.52	8.14	8.29	8.24	293.12	309.89	311.12	10.23	10.72	10.40
	Inception	148.57	**147.45**	**147.50**	10.48	**9.69**	**9.99**	263.18	**261.60**	**262.81**	9.07	**8.67**	**8.96**
	VGG	66.37	**66.01**	**66.14**	7.20	7.21	7.25	249.59	**248.52**	250.65	8.55	**8.31**	**8.45**
		CIFAR - 100											
PGD	**Resnet**	4.29	**4.29**	**3.94**	2.02	2.10	2.08	24.78	**24.67**	25.52	1.15	**1.11**	**1.10**
	Densenet	4.33	**4.33**	**4.12**	6.48	**6.48**	**6.47**	44.65	45.04	44.66	1.63	**1.62**	2.19
	Inception	4.31	**4.30**	**4.26**	6.07	**6.05**	**6.06**	44.09	44.36	44.12	1.41	**1.39**	**1.40**
	VGG	3.43	**3.41**	**3.37**	1.68	1.69	1.69	19.72	**19.66**	**19.68**	0.98	**0.98**	**0.97**
C&W	**Resnet**	9.96	**9.93**	**9.92**	0.61	0.62	0.62	22.84	24.30	**18.28**	1.15	1.18	1.18
	Densenet	22.78	**22.70**	**22.72**	0.58	**0.57**	0.59	26.47	26.46	26.62	1.61	**1.59**	**1.60**
	Inception	18.46	**18.31**	**18.15**	0.46	**0.46**	**0.46**	22.58	**22.50**	**22.42**	1.18	**1.14**	**1.15**
	VGG	7.68	**7.61**	**7.65**	0.39	**0.34**	**0.37**	21.35	**21.12**	**21.28**	0.98	**0.95**	**0.97**
StAdv	**Resnet**	10.51	**10.48**	**10.48**	1.72	**1.70**	**1.72**	129.56	**95.93**	**96.05**	0.98	1.01	1.03
	Densenet	23.50	**23.50**	23.54	1.07	1.09	1.08	38.44	40.64	40.80	1.34	1.41	1.36
	Inception	19.48	**19.34**	**19.34**	1.37	**1.30**	**1.34**	34.52	**34.31**	**34.47**	1.19	**1.18**	**1.16**
	VGG	8.70	**8.66**	**8.67**	0.94	**0.90**	**0.91**	32.73	**32.59**	32.87	1.12	**1.09**	**1.11**
		CIFAR - 10											
PGD	**Resnet**	4.94	**4.90**	**4.89**	2.87	**2.81**	**2.85**	25.66	**25.11**	**25.30**	1.55	**1.50**	**1.51**
	Densenet	4.83	**4.81**	**4.81**	6.94	**6.90**	**6.89**	45.21	45.22	45.26	1.99	**1.95**	**1.97**
	Inception	4.61	**4.55**	**4.57**	6.99	7.00	7.02	44.37	**44.11**	**44.27**	1.62	**1.60**	**1.58**
	VGG	3.94	**3.94**	3.99	2.15	**2.11**	**2.10**	20.61	**20.57**	**20.58**	1.23	**1.23**	1.25
C&W	**Resnet**	10.26	**10.20**	**10.19**	1.17	**1.15**	**1.16**	23.91	**23.87**	**23.90**	1.70	1.71	**1.70**
	Densenet	23.98	**23.97**	23.99	0.96	**0.89**	**0.91**	26.86	**26.81**	**26.81**	1.94	**1.90**	**1.93**
	Inception	19.85	**19.81**	**19.84**	0.88	**0.84**	**0.82**	23.07	**23.02**	**23.05**	1.39	1.40	1.41
	VGG	7.97	**7.92**	**7.95**	0.98	**0.96**	**0.92**	21.60	**21.48**	**21.51**	1.43	**1.38**	**1.41**
StAdv	**Resnet**	11.32	**11.28**	**11.30**	2.76	**2.70**	**2.74**	130.38	**130.10**	**130.29**	1.58	**1.52**	**1.54**
	Densenet	23.85	**23.84**	**23.84**	1.57	**1.55**	**1.50**	38.85	**38.81**	**38.79**	1.69	1.74	1.72
	Inception	19.93	**19.85**	**19.87**	1.85	**1.83**	**1.79**	34.74	**34.71**	35.13	1.54	**1.53**	**1.53**
	VGG	9.98	**9.91**	**9.93**	1.73	**1.70**	**1.72**	33.08	33.11	33.10	1.71	**1.69**	**1.71**

Best results are given in bold.

TABLE II

$\mathcal{L}_1$ ADVERSARIAL-TO-BENIGN DISTANCE USING TRANSFERABILITY OF ADVERSARIAL SAMPLES (ADV^2, ADVEDGE, AND ADVEDGE$^+$) ACROSS INTERPRETERS USING VARIOUS DNNS (ROW/COLUMN AS SOURCE/TARGET INTERPRETERS) FOR IMAGENET

			Grad			CAM			MASK			RTS		
			ADV^2	**AdvEdge**	**AdvEdge$^+$**	**ADV^2**	**AdvEdge**	**AdvEdge$^+$**	**ADV^2**	**AdvEdge**	**AdvEdge$^+$**	**ADV^2**	**AdvEdge**	**AdvEdge$^+$**
PGD	**Resnet**	**Grad**	0.081	**0.077**	**0.074**	0.246	**0.241**	**0.241**	0.578	**0.563**	**0.556**	0.092	**0.077**	**0.080**
		CAM	0.093	**0.092**	**0.091**	0.063	**0.063**	**0.062**	0.523	**0.523**	**0.523**	0.097	**0.097**	**0.096**
		MASK	0.117	**0.116**	**0.103**	0.246	**0.234**	0.262	0.545	**0.529**	**0.482**	0.099	**0.087**	**0.069**
		RTS	0.097	**0.096**	**0.096**	0.281	**0.273**	**0.273**	0.626	**0.613**	**0.617**	0.098	**0.097**	**0.097**
	Densenet	**Grad**	0.085	**0.082**	**0.079**	0.219	0.222	0.220	0.606	**0.596**	**0.575**	0.086	**0.072**	**0.073**
		CAM	0.102	**0.101**	**0.102**	0.093	**0.092**	**0.092**	0.573	**0.563**	**0.564**	0.100	**0.099**	**0.099**
		MASK	0.117	0.118	**0.115**	0.250	**0.246**	0.253	0.546	**0.523**	**0.511**	0.101	**0.086**	**0.086**
		RTS	0.104	**0.104**	**0.103**	0.262	**0.251**	**0.252**	0.567	**0.558**	**0.551**	0.110	**0.110**	**0.109**
	Inception	**Grad**	0.081	**0.081**	**0.078**	0.251	**0.249**	**0.245**	0.579	**0.565**	**0.556**	0.098	**0.082**	**0.085**
		CAM	0.101	**0.095**	**0.095**	0.071	**0.063**	**0.069**	0.526	0.531	**0.526**	0.104	**0.098**	**0.102**
		MASK	0.119	**0.117**	**0.112**	0.248	**0.239**	0.269	0.554	**0.535**	**0.484**	0.101	**0.089**	**0.078**
		RTS	0.097	0.104	0.103	0.283	**0.276**	**0.277**	0.631	**0.622**	**0.622**	0.099	**0.098**	0.105
	VGG	**Grad**	0.092	**0.087**	**0.084**	0.221	0.224	0.225	0.610	**0.604**	**0.576**	0.088	**0.076**	**0.073**
		CAM	0.102	0.107	0.106	0.097	0.100	**0.092**	0.573	**0.567**	**0.568**	0.102	**0.101**	0.103
		MASK	0.125	**0.119**	**0.124**	0.252	**0.247**	0.262	0.548	**0.525**	**0.516**	0.108	**0.086**	**0.090**
		RTS	0.107	**0.104**	0.109	0.268	**0.255**	**0.256**	0.570	**0.562**	**0.556**	0.110	**0.110**	0.115
C&W	**Resnet**	**Grad**	0.042	**0.041**	**0.041**	0.244	**0.240**	**0.242**	0.546	**0.545**	**0.542**	0.069	**0.069**	**0.069**
		CAM	0.050	0.060	**0.049**	0.328	**0.307**	0.329	0.573	**0.570**	0.574	0.038	**0.037**	**0.038**
		MASK	0.091	**0.089**	**0.089**	0.221	**0.220**	**0.219**	0.473	0.474	0.475	0.081	**0.081**	**0.081**
		RTS	0.092	**0.091**	**0.092**	0.308	0.310	0.311	0.639	**0.638**	**0.638**	0.086	**0.085**	**0.085**
	Densenet	**Grad**	0.045	**0.044**	**0.044**	0.241	**0.241**	**0.237**	0.562	**0.562**	**0.561**	0.056	**0.055**	**0.054**
		CAM	0.065	0.070	**0.065**	0.335	**0.333**	**0.335**	0.570	0.572	**0.570**	0.035	0.040	**0.035**
		MASK	0.092	**0.092**	0.093	0.220	**0.220**	**0.218**	0.479	**0.481**	**0.476**	0.080	**0.079**	**0.079**
		RTS	0.095	**0.094**	**0.094**	0.289	**0.286**	0.291	0.558	**0.558**	0.559	0.092	**0.091**	**0.091**
	Inception	**Grad**	0.052	**0.048**	0.053	0.246	**0.245**	**0.246**	0.564	0.565	**0.564**	0.061	**0.056**	**0.061**
		CAM	0.069	0.078	**0.065**	0.343	**0.334**	**0.335**	0.575	0.581	**0.574**	0.044	**0.040**	**0.040**
		MASK	0.097	0.100	**0.094**	0.223	**0.223**	**0.221**	0.484	**0.482**	**0.484**	0.088	**0.086**	**0.083**
		RTS	0.098	0.100	**0.098**	0.289	**0.288**	0.299	0.561	**0.560**	**0.560**	0.092	**0.092**	0.097
	VGG	**Grad**	0.042	**0.040**	**0.042**	0.252	**0.247**	**0.249**	0.551	**0.551**	**0.543**	0.077	**0.071**	**0.069**
		CAM	0.051	0.061	0.057	0.332	**0.316**	**0.332**	0.578	**0.570**	**0.572**	0.043	**0.041**	**0.040**
		MASK	0.100	**0.096**	**0.092**	0.227	**0.225**	**0.224**	0.476	**0.474**	**0.474**	0.082	**0.080**	**0.079**
		RTS	0.101	**0.099**	**0.093**	0.310	0.317	0.311	0.646	**0.647**	**0.644**	0.086	**0.086**	0.090
StAdv	**Resnet**	**Grad**	0.036	**0.035**	**0.035**	0.259	**0.229**	**0.231**	0.550	**0.539**	**0.540**	0.032	**0.031**	**0.031**
		CAM	0.078	**0.076**	**0.077**	0.141	**0.114**	**0.115**	0.542	**0.536**	**0.537**	0.023	**0.020**	**0.020**
		MASK	0.084	0.086	0.086	0.246	**0.243**	**0.246**	0.511	**0.501**	**0.502**	0.074	**0.073**	**0.071**
		RTS	0.079	**0.078**	**0.078**	0.255	**0.248**	**0.249**	0.550	**0.548**	**0.547**	0.022	**0.021**	**0.021**
	Densenet	**Grad**	0.041	**0.040**	**0.040**	0.250	**0.221**	**0.222**	0.547	**0.538**	**0.539**	0.034	0.036	0.036
		CAM	0.088	**0.085**	**0.085**	0.172	**0.151**	**0.151**	0.545	**0.541**	**0.541**	0.062	0.064	**0.062**
		MASK	0.085	**0.085**	**0.084**	0.229	**0.216**	**0.219**	0.493	**0.485**	**0.489**	0.067	**0.067**	0.068
		RTS	0.082	**0.081**	**0.081**	0.241	**0.235**	**0.235**	0.543	**0.539**	**0.540**	0.028	**0.020**	**0.020**
	Inception	**Grad**	0.039	**0.038**	0.041	0.267	**0.230**	**0.238**	0.558	**0.544**	**0.546**	0.041	**0.039**	**0.038**
		CAM	0.084	0.085	**0.079**	0.148	**0.117**	**0.122**	0.546	**0.544**	**0.545**	0.031	**0.028**	0.035
		MASK	0.088	0.092	**0.088**	0.250	**0.249**	**0.249**	0.511	**0.509**	**0.511**	0.078	0.079	0.071
		RTS	0.085	**0.084**	**0.080**	0.258	**0.252**	**0.252**	0.555	**0.553**	**0.551**	0.023	0.029	0.024
	VGG	**Grad**	0.046	**0.042**	**0.042**	0.253	**0.229**	**0.228**	0.547	**0.538**	**0.545**	0.038	**0.036**	**0.038**
		CAM	0.090	0.092	**0.088**	0.178	**0.156**	**0.160**	0.545	**0.541**	**0.541**	0.063	**0.060**	**0.061**
		MASK	0.092	**0.087**	**0.090**	0.232	**0.225**	**0.224**	0.495	**0.492**	**0.489**	0.071	**0.068**	**0.068**
		RTS	0.085	0.089	**0.085**	0.245	**0.240**	**0.238**	0.552	**0.545**	**0.541**	0.037	**0.030**	**0.031**

Best results are given in bold.

Fig. 5. Noise rate of adversarial inputs generated by ADV^2, AdvEdge, and AdvEdge$^+$ on ResNet-50, DenseNet-169, VGG-16 and Inception-V3 for ImageNet dataset with three different attack frameworks: PGD, C&W, and StAdv.

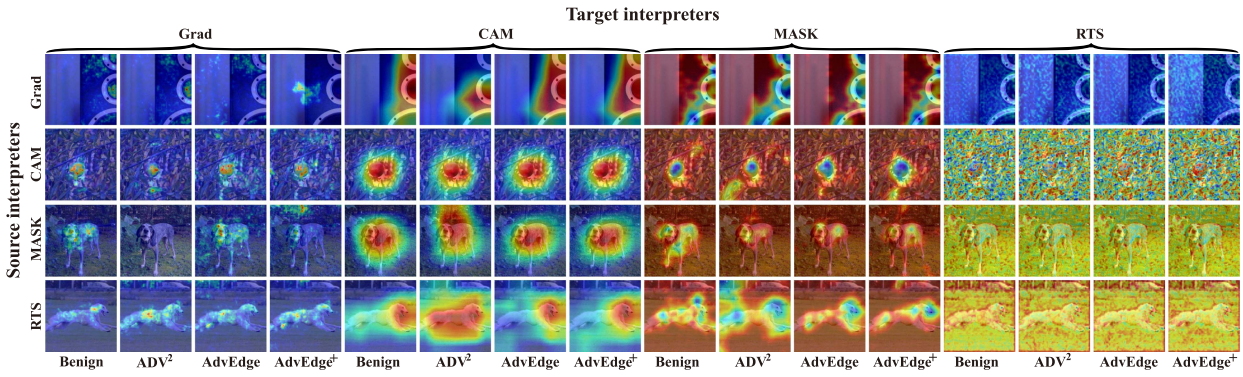

Fig. 6. Transferability of adversarial attribution maps of across different interpreters on ResNet-50 using ADV^2, AdvEdge, and AdvEdge$^+$ with PGD. The results show that AdvEdge and AdvEdge$^+$ provide highly similar attribution maps to benign cases.

similar performances. We selected MS, BS, and AT defenses because of their high performance in the literature. For AT, we generated adversarial samples using a PGD attack with three different iterations (50, 100, and 150), resulting in a triple-sized adversarial training dataset.

Using the ImageNet dataset, our attack success rate was higher when applying the PGD framework compared to C&W and StAdv. For example, in MS, the attack success rates were 35.2%, 23.8%, and 14.4% when applying PGD, C&W, and StAdv, respectively. In BS, our attack performed similarly to MS. However, in AT, our attack achieved higher attack success rates: 42.5%, 50.1%, and 33.3%, respectively.

Using the CIFAR-100 and CIFAR-10 datasets, our attack achieved similar attack success rates, except for AT. In AT, our attack achieved 55.0%, 55.7%, and 31.9% when applying PGD, C&W, and StAdv frameworks. From the results, we can conclude that even though our attack perturbs a very small amount of noise, the attack success rate is good. Our attack shows a lower attack success rate in MS and BS because these defense techniques are designed to filter noise. In the case of AT, our results are comparatively higher than the results against MS and BS. Full results are in Appendix D (Table XI), available online.

VI. Discussion

Our analysis suggests that AdvEdge and AdvEdge$^+$ could encounter challenges, as shown in Fig. 3, mainly when the sample is close to a decision boundary or when the interpretation map is scattered throughout the entire image. These challenges are also shown in Fig. 7, which shows examples of outliers in various attack frameworks on ResNet and DenseNet. Our work suggests new defense strategies and underlines the need for

TABLE III
RESULTS OF A REAL APPLICATION EXPERIMENT IN WHICH A BLACK-BOX MODEL IS ATTACKED USING THE TRANSFER LEARNING TECHNIQUE

		Dataset	Accuracy	Attack method	# samples	Attack success rate	Avg. number of queries	Adv. IoU scores (average)
T.M.	EfficientNetb3	Brain Tumor MRI [42]	0.99	AdvEdge + MGA [43]	200	0.90	365.15	0.86
S.M.	ResNet50	Brain Tumor MRI [44]	0.96					0.90

T.M., S.M., and Adv. stand for Target Model, Surrogate Model, and Adversarial, respectively.

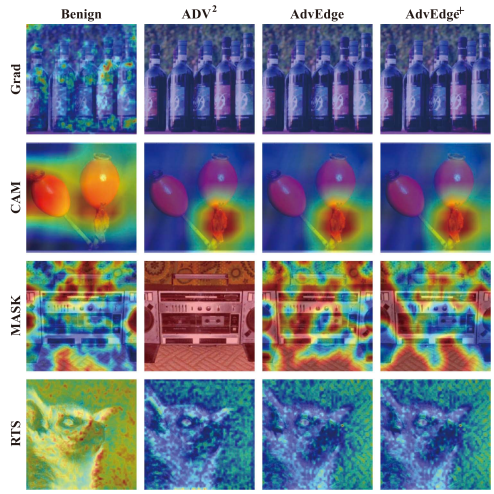

Fig. 7. Outliers of adversarial inputs with interpretations similar to benign. Grad and MASK are based on StAdv and PGD, respectively (DenseNet-169). CAM and RTS interpreters are based on StAdv and C&W, respectively (Resnet-50).

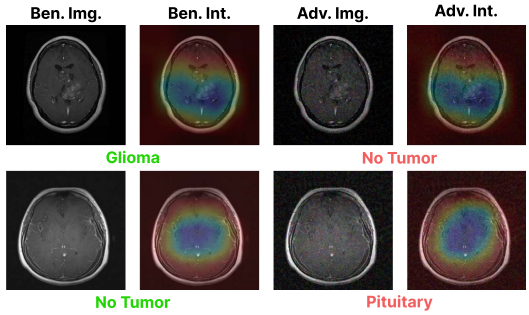

Fig. 8. Example images for the real application case in which adversarial examples are generated against the surrogate model and tested on the target model.

continuous development in this area, keeping pace with advances in adversarial attack techniques.

A. Real-World Application

As an example of real-life scenarios, we evaluate the attacks against a brain tumor classification task using the open-source datasets [42] and [44]. The datasets contain MRI images for four categories.

Threat Model: To make the scenario seem realistic, we assume a *black-box attack* with the target being the black-box IDLS, from which we can only obtain the model output. The adversary has neither knowledge nor access to the victim classifier nor its interpreter. The attacker must rely on general adversarial techniques and any public information available. The adversarial samples are crafted without direct knowledge of the model's architecture or parameters, making this the most challenging and realistic attack scenario. We will consider the case to highlight the seriousness of our attack and the potential implications for the real world. We assume that the adversary has access to medical datasets comparable to those the target IDLS uses for training. This is reasonable because probing the target IDLS requires certain input specifications.

Attack Implementation: We select two models with different architectures to serve as target and surrogate models. First, we train both models, the target and surrogate models, using different but similar datasets, (i.e., [42] and [44], respectively). Using transferability, the surrogate model is used to create adversarial samples that are then sent to the target model. Table III shows the performance of the models and AdvEdge against the target model. To increase the transferability of adversarial samples, we employ a genetic algorithm-based approach [43] to improve the attack success rate. Using 200 samples, Table III shows that AdvEdge achieves an attack success rate of 0.90 and an IoU score of 0.86. Fig. 8 shows example images that generated similar interpretations with different classification outputs.

B. Potential Countermeasures

Our analysis shows that different interpretation models utilize different behavioral aspects of DNN models. This suggests the use of an ensemble of interpretation techniques, i.e., leveraging the strengths of multiple interpreters, to defend against interpretation-based attacks. However, grouping interpretations using the ensemble technique can be challenging because interpretations by different interpreters may not always match. We address this by finding cost-effective countermeasures that improve defense while remaining computationally efficient.

Interpretations Ensemble Adversarial Detector: We propose a novel multiple-interpreter-based detector that utilizes multiple interpreters to check whether a given input is adversarial or benign. For our experiment, we generated adversarial samples based on one interpreter and produce attribution maps of those samples via two interpreters including the target interpreter. For example, for adversarial samples that are generated with the Grad interpreter, we obtain the attribution maps of those samples from both Grad and CAM interpreters. We adopt the same process by targeting CAM and applying Grad as a secondary interpreter. The generated attribution maps are based on single-channel; therefore, we stacked single-channel attribution maps of two interpreters to convert them into benign and adversarial two-channel data. 2,000 benign and 2,000 adversarial samples are produced for each experiment. We adopt the pre-trained DNN model VGG-16 [37] to extract feature vectors from convolutional layers and we adopt the gradient-boosting classifier as the final layer instead of the fully connected layer. The

TABLE IV
RESULTS OF THE TWO-INTERPRETATION-BASED DETECTOR

2-channel samples				
	Grad	**CAM**	**MASK**	**RTS**
Grad	-	-	-	-
CAM	0.896	-	-	-
MASK	0.869	0.866	-	-
RTS	0.762	0.716	0.701	-
3-channel samples				
Grad	-	-	-	-
CAM	0.890	-	-	-
MASK	0.911	0.894	-	-
RTS	0.744	0.712	0.714	-

The upper part relates to 2-channel samples and the lower part shows the results of 3-channel samples.

TABLE V
RESULTS OF THE THREE-INTERPRETATION-BASED DETECTOR

Combinations of interpreters	**Accuracy**
Grad, CAM, MASK	0.975
Grad, CAM, RTS	0.861
Grad, MASK, RTS	0.838
CAM, MASK, RTS	0.848

main reason for the approach is that the benign and adversarial attribution maps are highly similar and difficult to classify. The gradient-boosting classifier combines several weak learning models to form a strong predictive model. Since each attribution map is a one-channel image, we replaced the input and output layers of VGG-16 and trained only those layers. In another setting, we also created a third channel by multiplying the two attribution maps collected from two interpreters. Table IV show the results of both cases. The detector could help improve the adversarial detection process. We examine adopting three different interpreters by stacking their attribution maps into a 3-channel input for the detector. Table V shows the detector results based on the combination of three interpreters.

By leveraging several interpretations, the detector demonstrates promising results in improving the system robustness. We also note that more research is needed to assess computational complexity and resource requirements when deploying such a detector on larger datasets and real-world systems.

Interpretation-Based Robust Training: The defense technique is based on the method in [45]. The technique considers a generic form of L_1 2-class interpretation difference:

$$\mathbb{D}_{L_1}(x,\hat{x}) = (1/2)(\|g_y(x,f) - g_y(\hat{x},f)\|_1 + \|g_{y_t}(x,f) - g_{y_t}(\hat{x},f)\|_1).$$

This creates a perturbation-independent lower bound for any adversarial attack that makes it hard to fool a classifier and evades the interpretation discrepancy. Training a classifier against the worst-case interpretation difference is recommended by the min-max optimization problem

$$\min_{\theta} \mathbb{E}_x \left[f_{train}(\theta; x, y) + \gamma \mathbb{D}_{worst}(x, \hat{x}) \right],$$

where f_{train} is the training loss (i.e., cross-entropy loss), $\mathbb{D}_{worst}$ denotes a measurement of the highest interpretation difference between benign and adversarial samples x and $\hat{x}$, and γ balances

TABLE VI
RESULTS OF INTERPRETATION-BASED ROBUST TRAINING WITH DIFFERENT PERTURBATION SIZE ϵ

ϵ	*0*	*2/255*	*4/255*	*6/255*	*8/255*	*9/255*	*10/255*
Testing Accuracy							
NT	**0.785**	0.190	0.090	0.080	0.085	0.085	0.080
AT	0.685	**0.565**	**0.465**	**0.335**	**0.280**	**0.265**	**0.220**
Attack against interpretability							
NT		0.542	0.132	-0.06	-0.100	-0.083	-0.183
AT		**0.870**	**0.782**	**0.736**	**0.703**	**0.706**	**0.676**

NT stands for Normal Training and AT denotes Adversarial Training. Testing accuracy is based on the robustness of classifier (upper part) and the lower part provides the results of Kendall's Tau order rank correlation in the similarity of attribution maps.

the accuracy and model interpretability. For the experiment, we adopted PGD attack framework to generate adversarial samples. The results of the robust training are provided in Table VI. In the table, the results are produced by the Wide-ResNet [46] model with coupled CAM interpreter on CIFAR-10 [35] dataset. Interpretability results are calculated using Kendall's Tau order rank correlation between the original and adversarial interpretation maps. As shown in the Table VI, the adversarial-trained model provided high classification and interpretation robustness compared to the normal-trained model.

While the results indicate high classification and interpretation robustness, we recognize the importance of evaluating the feasibility of deploying this defense in practical scenarios. Successful implementation requires considering factors such as the availability of diverse training data, computational cost, generalization capabilities, and trade-offs among robustness, accuracy, and interpretability. Finding the right balance is crucial for practical deployment.

Other Cost-Effective Defense Techniques: Cost-effective defense techniques offer practical and efficient ways to enhance robustness. These techniques (e.g., data augmentation, gradient masking, jpeg compression, and feature squeezing) balance effectiveness and computational efficiency well, making them suitable for real-world applications with limited resources and time constraints. Although no single defense method can ensure complete robustness, employing a combination of such techniques can significantly strengthen model defense against common attacks. We provide a result on such defenses, i.e., median smoothing, bit squeezing, and typical adversarial training in Appendix D (Table XI), available online. We note that there exist other potential countermeasures and strategies to enhance the robustness of the system. For future work, we aim to broaden our investigation to include other defenses, such as [47].

VII. CONCLUSION

This paper proposes two attack methods (AdvEdge and AdvEdge$^+$) to improve the adversarial attacks on interpretable deep learning systems (IDLSes). These methods use edge information of the image inputs to optimize the ADV^2 attack, which provides adversarial samples to mislead the target DNN models and their coupled interpretation models simultaneously. By empirically examining three large datasets in four distinct DNN architectures by adopting three different attack frameworks (PGD, StAdv, and C&W), we showed the validity and

efficacy of AdvEdge and AdvEdge$^+$ attacks. We compared our findings with four different types of interpreters (i.e., Grad, CAM, MASK, and RTS). The results indicated that AdvEdge and AdvEdge$^+$ effectively produce adversarial samples capable of misleading the DNN models and their interpreters. We also showed that the transferability of attribution maps generated by our attack methods provides significantly better results than the existing methods. The results confirmed that the interpretability of IDLSes provides a limited sense of security in the decision-making process.

REFERENCES

[1] E. Abdukhamidov, M. Abuhamad, F. Juraev, E. Chan-Tin, and T. AbuHmed, "AdvEdge: Optimizing adversarial perturbations against interpretable deep learning," in *Proc. Int. Conf. Comput. Data Soc. Netw.*, D. Mohaisen and R. Jin, Eds. Cham:Springer International Publishing, 2021, pp. 93–105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014, *arXiv:1409.3215*.

[4] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," 2017, *arXiv:1710.09829*.

[5] Q. Zhang, Y. N. Wu, and S.-C. Zhu, "Interpretable convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8827–8836.

[6] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," 2017, *arXiv:1705.07857*.

[7] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3429–3437.

[8] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014, *arXiv:1412.6806*.

[9] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 618–626.

[10] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2921–2929.

[11] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 427–436.

[12] N. Liu, H. Yang, and X. Hu, "Adversarial detection with model interpretation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1803–1811.

[13] G. Tao, S. Ma, Y. Liu, and X. Zhang, "Attacks meet interpretability: Attribute-steered detection of adversarial samples," 2018, *arXiv:1810.11580*.

[14] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *Proc. 29th USENIX Secur. Symp.*, 2020, Art. no. 94.

[15] V. Duddu and A. Boutet, "Inferring sensitive attributes from model explanations," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 416–425.

[16] S. Fang and A. Choromanska, "Backdoor attacks on the DNN interpretation system," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 561–570. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/19935

[17] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[18] E. Abdukhamidov, F. Juraev, M. Abuhamad, and T. Abuhmed, "Black-box and target-specific attack against interpretable deep learning systems," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2022, pp. 1216–1218.

[19] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2016, *arXiv:1611.02770*.

[20] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016, *arXiv:1605.07277*.

[21] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie, and S.-N. Lim, "Enhancing adversarial example transferability with an intermediate level attack," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4733–4742.

[22] C. Xie et al., "Improving transferability of adversarial examples with input diversity," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2730–2739.

[23] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*.

[24] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1765–1773.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[26] A. Kurakin et al., "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*.

[27] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.

[28] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," 2018, *arXiv:1801.02612*.

[29] I. Sobel, R. Duda, P. Hart, and J. Wiley, "Sobel-feldman operator." Accessed: Dec. 20, 2023. [Online]. Available: https://www.researchgate.net/profile/Irwin-Sobel/publication/285159837

[30] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One*, vol. 10, no. 7, 2015, Art. no. e0130140.

[31] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: Removing noise by adding noise," 2017, *arXiv:1706.03825*.

[32] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks," 2017, *arXiv:1711.06104*.

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, Springer, 2015, pp. 234–241.

[34] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do ImageNet classifiers generalize to ImageNet?," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 5389–5400.

[35] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, Tech. Rep. TR-2009, 2009.

[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[39] J. Heo, S. Joo, and T. Moon, "Fooling neural network interpretations via adversarial model manipulation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2921–2932.

[40] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proc. AAAI/ACM Conf. AI Ethics Soc.*, 2020, pp. 180–186.

[41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[42] M. Nickparvar, "Brain tumor MRI dataset," 2021. [Online]. Available: https://www.kaggle.com/dsv/2645886

[43] L. Wang, K. Yang, W. Wang, R. Wang, and A. Ye, "MGAAttack: Toward more query-efficient black-box attack by microbial genetic algorithm," in *Proc. 28th ACM Int. Conf. Multimedia*, 2020, pp. 2229–2236.

[44] S. Bhuvaji, A. Kadam, P. Bhumkar, S. Dedge, and S. Kanchan, "Brain tumor classification (MRI)," 2020. [Online]. Available: https://www.kaggle.com/dsv/1183165

[45] A. Boopathy et al., "Proper network interpretability helps adversarial robustness in classification," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 1014–1023.

[46] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[47] M. Huai, J. Liu, C. Miao, L. Yao, and A. Zhang, "Towards automating model explanations with certified robustness guarantees," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 6935–6943.

Eldor Abdukhamidov received the BS degree from Inha University in Tashkent, in 2019. He is currently working toward the master's and PhD degrees in computer science and engineering with Sungkyunkwan University, Suwon, South Korea. His main research interests include image processing, machine learning and information security.

Mohammed Abuhamad received the PhD degree in computer science from the University of Central Florida, in 2020, and the PhD degree in electrical and computer engineering from INHA University, in 2020. He is currently an assistant professor of computer science with Loyola University Chicago. His research interests include deep learning-based applications in information security, software and mobile/IoT security, and adversarial machine learning.

Simon S. Woo received the BS degree in electrical engineering from the University of Washington (UW), Seattle, the MS degree in electrical and computer engineering from the University of California, San Diego (UCSD), and the MS and PhD degrees in computer science from the University of Southern California (USC), Los Angeles. He was a member of technical staff (technologist) for 9 years with the NASA's Jet Propulsion Lab (JPL), Pasadena, California, conducting research in satellite communications, networking and cybersecurity areas. Also, He worked with Intel Corporation and Verisign Research Laboratory. Since 2017, he was a tenure-track assistant professor with SUNY, South Korea and a research assistant professor with Stony Brook University. Now, he is a tenure-track assistant professor with the SKKU Institute for Convergence and Department of Applied Data Science and Software in Sungkyunkwan University, Suwon, Korea.

Eric Chan-Tin received the BA degree in computer science and mathematics from Macalester College, and the PhD degree in computer science from the University of Minnesota. He is currently an associate professor with the Department of Computer Science, Loyola University Chicago. His research focuses broadly on network security, distributed systems, privacy, and anonymity.

Tamer Abuhmed received the PhD degree in information and telecommunication engineering from Inha University, in 2012. He is currently an associate professor with the College of Computing and Informatics, Sungkyunkwan University, South Korea. His research interests include information security, software and system security, adversarial machine learning, and expert systems.