DECOY-ENHANCED SALIENCY MAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Saliency methods can make deep neural network predictions more interpretable by identifying a set of critical features in an input sample, such as pixels that contribute most strongly to a prediction made by an image classifier. Unfortunately, recent evidence suggests that many saliency methods poorly perform, especially in situations where gradients are saturated, inputs contain adversarial perturbations, or predictions rely upon inter-feature dependence. To address these issues, we propose a framework that improves the robustness of saliency methods by following a two-step procedure. First, we introduce a perturbation mechanism that subtly varies the input sample without changing its intermediate representations. Using this approach, we can gather a corpus of perturbed data samples while ensuring that the perturbed and original input samples follow the same distribution. Second, we compute saliency maps for the perturbed samples and propose a new method to aggregate saliency maps. With this design, we offset the gradient saturation influence upon interpretation. From a theoretical perspective, we show that the aggregated saliency map not only captures inter-feature dependence but, more importantly, is robust against previously described adversarial perturbation methods. Following our theoretical analysis, we present experimental results suggesting that, both qualitatively and quantitatively, our saliency method outperforms existing methods, in a variety of applications.

1 INTRODUCTION

Deep neural networks (DNNs) deliver remarkable performance in an increasingly wide range of application domains, but they often do so in an inscrutable fashion, delivering predictions without accompanying explanations. In a practical setting such as automated analysis of pathology images, if a patient sample is classified as malignant, then the physician will want to know which parts of the image contribute to this diagnosis. Thus, in general, a DNN that delivers interpretations alongside its predictions will enhance the credibility and utility of its predictions for end users (Lipton, 2016).

In this paper, we focus on a popular branch of explanation methods, often referred to as *saliency methods*, which aim to find input features (*e.g.,* image pixels or words) that strongly influence the network predictions (Simonyan et al., 2013; Selvaraju et al., 2016; Binder et al., 2016; Shrikumar et al., 2017; Smilkov et al., 2017; Sundararajan et al., 2017; Ancona et al., 2018). Saliency methods typically rely on back-propagation from the network's output back to its input to assign a saliency score to individual features so that higher scores indicate higher importance to the output prediction. Despite attracting increasing attention, saliency methods suffer from several fundamental limitations:

- **Gradient saturation** (Sundararajan et al., 2017; Shrikumar et al., 2017; Smilkov et al., 2017) may lead to the problem that the gradients of important features have small magnitudes, breaking down the implicit assumption that important features, in general, correspond to large gradients. This issue can be triggered when the DNN outputs are flattened in the vicinity of important features.
- **Importance isolation** (Singla et al., 2019) refers to the problem that gradient-based saliency methods evaluate the feature importance in an isolated fashion, implicitly assuming that the other features are fixed.
- **Perturbation sensitivity** (Ghorbani et al., 2017; Kindermans et al., 2017; Levine et al., 2019) refers to the observation that even imperceivable, random perturbations or a simple shift transformation of the input data may lead to a large change in the resulting saliency scores.

In this paper, we tackle these limitations by proposing a decoy-enhanced saliency score. At a high level, our method generates the saliency score of an input by aggregating the saliency scores of multiple perturbed copies of this input. Specifically, given an input sample of interest, our method first generates a population of perturbed samples, referred to as *decoys*, that perfectly mimic the neural network's intermediate representation of the original input. These decoys are used to model the variation of an input sample originating from either sensor noise or adversarial attacks. The decoy construction procedure draws inspiration from the *knockoffs*, proposed recently by Barber & Candès (2015) in the setting of error-controlled feature selection, where the core idea is to generate knockoff features that perfectly mimic the empirical dependence structure among the original features.

In brief, the current paper makes three primary contributions. First, we propose a framework to perturb input samples to produce corresponding decoys that preserve the input distribution, in the sense that the intermediate representations of the original input data and the decoys are indistinguishable. We formulate decoy generation as an optimization problem, applicable to diverse deep neural network architectures. Second, we develop a decoy-enhanced saliency score by aggregating the saliency maps of generated decoys. By design, this score naturally offsets the impact of gradient saturation. From a theoretical perspective, we show how the proposed score can simultaneously reflect the joint effects of other dependent features and achieve robustness to adversarial perturbations. Third, we demonstrate empirically that the decoy-enhanced saliency score outperforms existing saliency methods, both qualitatively and quantitatively, on three real-world applications. We also quantify our method's advantage over existing saliency methods in terms of robustness against various adversarial attacks.

2 Related work

A variety of saliency methods have been proposed in the literature. Some, such as edge detectors and Guided Backpropagation (Springenberg et al., 2014) are independent from the predictive model (Nie et al., 2018; Adebayo et al., 2018). [1] Others are designed only for specific architectures (*i.e.,* Grad-CAM (Selvaraju et al., 2016) for CNNs, DeConvNet for CNNs with ReLU activations (Zeiler & Fergus, 2014)). In this paper, instead of exhaustively evaluating all saliency methods, we apply our method to the three saliency methods that do depend on the predictor (*i.e.,* passing the sanity checks in Adebayo et al. (2018) and Sixt et al. (2020)) and are applicable to diverse DNN architectures:

- The **vanilla gradient** method (Simonyan et al., 2013) simply calculates the gradient of the class score with respect to the input $\mathbf{x}$, which is defined as $E_{grad}(\mathbf{x}; F^c) = \nabla_{\mathbf{x}} F^c(\mathbf{x})$.
- The **SmoothGrad** method (Smilkov et al., 2017) seeks to reduce noise in the saliency map by averaging over explanations of the noisy copies of an input, defined as $E_{sg}(\mathbf{x}; F^c) = \frac{1}{N}\sum_{i=1}^{N} E_{grad}(\mathbf{x} + g_i; F^c)$ with noise vectors $g_i \sim N(0, \sigma^2)$.
- The **integrated gradient** method [2] (Sundararajan et al., 2017) starts from a baseline input $\mathbf{x}^0$ and sums over the gradient with respect to scaled versions of the input ranging from the baseline to the observed input, defined as $E_{ig}(\mathbf{x}; F^c) = (\mathbf{x} - \mathbf{x}^0) \times \int_0^1 \nabla_{\mathbf{x}} F^c(\mathbf{x}^0 + \alpha(\mathbf{x} - \mathbf{x}^0)) d\alpha$.

We do not empirically compare to several other categories of methods. *Counterfactual-based methods* work under the same setup as saliency methods, providing explanations for the predictions of a pre-trained DNN model (Sturmfels et al., 2020). These methods identify the important subregions within an input image by perturbing the subregions (by adding noise, rescaling (Sundararajan et al., 2017), blurring (Fong & Vedaldi, 2017), or inpainting (Chang et al., 2019)) and measuring the resulting changes in the predictions (Ribeiro et al., 2016; Lundberg & Lee, 2017; Chen et al., 2018; Fong & Vedaldi, 2017; Dabkowski & Gal, 2017; Chang et al., 2019; Yousefzadeh & O'Leary, 2019; Goyal et al., 2019). Although these methods do identify meaningful subregions in practice, they exhibit several limitations. First, counterfactual-based methods implicitly assume that regions containing the object most contribute to the prediction (Fan et al., 2017). However, Moosavi-Dezfooli et al. (2017) showed that counterfactual-based methods are also vulnerable to adversarial attacks, which force these methods to output unrelated background rather than the meaningful objects as important subregions.

[1] Sixt et al. (2020) shows that LRP (Binder et al., 2016) is independent of the parameters of certain layers.

[2] Ancona et al. (2018) shows that $input \odot gradient$ and DeepLIFT (Shrikumar et al., 2017) are strongly related to the integrated gradient. As such, we only select the integrated gradient.

Second, the counterfactual images may be potentially far away from the training distribution, causing ill-defined classifier behavior (Burns et al., 2019; Hendrycks & Dietterich, 2019).

In addition to these limitations, counterfactual-based methods and our decoy-based method are fundamentally different in three ways. First, the former seeks the minimum set of features to exclude in order to minimize the prediction score or to include in order to maximize the prediction score (Fong & Vedaldi, 2017), whereas our approach aims to characterize the influence of each feature on the prediction score. Second, counterfactual-based methods explicitly consider the decision boundary by comparing each image to the closest image on the other side of the boundary. In contrast, the proposed method only considers the decision boundary implicitly by calculating the gradient's variants. Third, unlike counterfactual images, which could potentially be out-of-distribution, decoys are plausibly constructed in the sense that their intermediate representations are indistinguishable from the original input data by design. Because of these limitations and differences, we do not compare our method with counterfactual-based methods.

In addition to saliency methods and counterfactual-based methods, several other types of interpretation methods have been proposed that either aim for a different goal or have a different setup. For example, recent research (*e.g.,* Ribeiro et al. (2016); Lundberg & Lee (2017); Chen et al. (2018; 2019b)) designed techniques to explain a black-box model, where the model's internal weights are inaccessible. Koh & Liang (2017) and some follow-up work (Yeh et al., 2018; Koh et al., 2019) tried to find the training points that are most influential for a given test sample. Some other efforts have been made to train a more interpretable DNN classifier (Fan et al., 2017; Zołna et al., 2019; Alvarez-Melis & Jaakkola, 2018; Toneva & Wehbe, 2019), synthesize samples that represent the model predictions (Ghorbani et al., 2019; Chen et al., 2019a)), or identifying noise-tolerant features (Ikeno & Hara, 2018; Schulz et al., 2020). However, due to the task and setup differences, we do not consider these methods in this paper.

3 METHODS

3.1 PROBLEM SETUP

Consider a multi-label classification task in which a pre-trained neural network model implements a function F: $\mathbb{R}^d \mapsto \mathbb{R}^C$ that maps from the given input $\mathbf{x} \in \mathbb{R}^d$ to C predicted classes. The score for each class $c \in \{1, \cdots, C\}$ is $F^c(\mathbf{x})$, and the predicted class is the one with maximum score, *i.e.,* $\arg\max_{c\in\{1,\cdots,C\}} F^c(\mathbf{x})$. A *saliency method* aims to assign to each feature a *saliency score*, encoded in a saliency map $E(\mathbf{x}; F^c) : \mathbb{R}^d \mapsto \mathbb{R}^d$, in which the features with higher scores represent higher "importance" relative to the final prediction.

Given a pre-trained neural network model F with L layers, an input $\mathbf{x}$, and a saliency method E such that $E(\mathbf{x}; F)$ is a saliency map of the same dimensions as $\mathbf{x}$, the proposed scores can be obtained in two steps: generating decoys and aggregating the saliency maps of the decoys.

3.2 DECOY DEFINITION

Say that $F_\ell : \mathbb{R}^d \mapsto \mathbb{R}^{d_\ell}$ is the function instantiated by the given network, which maps from an input $\mathbf{x} \in \mathbb{R}^d$ to its intermediate representation $F_\ell(\mathbf{x}) \in \mathbb{R}^{d_\ell}$ at layer $\ell \in \{1, 2, \cdots, L\}$. A vector $\tilde{\mathbf{x}} \in \mathbb{R}^d$ is said to be a *decoy* of $\mathbf{x} \in \mathbb{R}^d$ at a specified layer ℓ if the following swappable condition is satisfied:

$$F_\ell(\mathbf{x}) = F_\ell(\mathbf{x}_{\text{swap}(\tilde{\mathbf{x}},\mathcal{K})}), \text{ for swappable features } \mathcal{K} \subset \{1, \cdots, d\}. \tag{1}$$

Here, the swap$(\tilde{\mathbf{x}}, \mathcal{K})$ operation swaps features between $\mathbf{x}$ and $\tilde{\mathbf{x}}$ based on the elements in $\mathcal{K}$. In this work, $\mathcal{K}$ represents a small meaningful feature set, which represents a small region/segment in an image or a group of words (embeddings) in a sentence. Take an image recognition task for example. Assume $\mathcal{K} = \{10\}$ and $\tilde{\mathbf{x}}$ is a zero matrix, then $\mathbf{x}_{\text{swap}(\tilde{\mathbf{x}},\mathcal{K})}$ indicates a new image that is identical to $\mathbf{x}$ except that the tenth pixel is set to zero. Using the swappable condition, we aim to ensure that the original image $\mathbf{x}$ and its decoy $\tilde{\mathbf{x}}$ are indistinguishable in terms of the intermediate representation at layer ℓ. Note in particular that the construction of decoys relies solely on the first ℓ layers of the neural network $F_1, F_2, \cdots, F_\ell$ and is independent of the succeeding layers $F_{\ell+1}, \cdots, F_L$. As such, $\tilde{\mathbf{x}}$ is conditionally independent of the classification task $F(\mathbf{x})$ given the input $\mathbf{x}$; *i.e.,* $\tilde{\mathbf{x}} \perp\!\!\!\perp F(\mathbf{x}) | \mathbf{x}$.

3.3 DECOY GENERATION

To identify decoys satisfying the swappable condition, we solve the following optimization problem:

$$\begin{aligned} \text{maximize}_{\tilde{\mathbf{x}} \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]^d} \quad & \left\| ((\tilde{\mathbf{x}} - \mathbf{x}) \cdot s)^+ \right\|_1 , \\ \text{s.t.} \quad & \begin{cases} \|F_\ell(\tilde{\mathbf{x}}) - F_\ell(\mathbf{x})\|_\infty \leq \epsilon, \\ (\tilde{\mathbf{x}} - \mathbf{x}) \circ (1 - \mathcal{M}) = 0 \end{cases} \end{aligned} \tag{2}$$

Here, $(\cdot)^+ = \max(\cdot, 0)$, and the operators $\|\cdot\|_1$ and $\|\cdot\|_\infty$ correspond to the L_1 and L_∞ norms, respectively. $\mathcal{M} \in \{0,1\}^d$ is a specified binary mask. And the value of each feature in the decoy $\tilde{\mathbf{x}}$ is restricted to lie in a legitimate value range *i.e.,* $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$ (*e.g.,* the pixel value should lie in [0, 255]). We impose the constraint $\|F_\ell(\tilde{\mathbf{x}}) - F_\ell(\mathbf{x})\|_\infty \leq \epsilon$, which ensures that the generated decoy satisfies the swappable condition described in Eqn. 1. It should be noted that we take $\tilde{\mathbf{x}}$ and $\mathbf{x}$ to be indistinguishable except for the swappable features indicated by the mask (*i.e.,* $\mathbf{x}_{swap(\tilde{\mathbf{x}}, \mathcal{K})} = \tilde{\mathbf{x}}$).

As is shown later in Section 3.4, our decoy-enhanced saliency score is defined to capture the empirical range of the decoy saliencies. Here, we first need to estimate the upper/lower ends of the legitimate decoys. To achieve this, in Eqn. 2, we maximize the deviation between $\tilde{\mathbf{x}}$ and $\mathbf{x}$ from both the positive and negative directions, *i.e.,* $s = +1$ and $s = -1$. By using this objective function, for each mask $\mathcal{M}$, we can compute two decoys—one for the positive deviation (*i.e.,* $s = +1$) and the other for the negative one (*i.e.,* $s = -1$). For details of the optimization procedure, see Section A6.

3.4 DECOY-ENHANCED SALIENCY SCORES

Given an input sample $\mathbf{x}$ and a swappable patch with size K, we obtain n unique masks by sliding the swappable patch across the input with a certain stride. Then, we can generate $2n$ decoys for that sample, denoted $\left\{\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \cdots, \tilde{\mathbf{x}}^{2n}\right\}$. For these decoys, we can then apply a given saliency method E to yield the corresponding decoy saliency maps $\left\{E(\tilde{\mathbf{x}}^1; F), E(\tilde{\mathbf{x}}^2; F), \cdots, E(\tilde{\mathbf{x}}^{2n}; F)\right\}$. With these decoy saliency maps in hand, for each feature $\mathbf{x}_i$ in $\mathbf{x}$, we can characterize its saliency score variation by using a population of saliency scores $\tilde{E}_i = \left\{E(\tilde{\mathbf{x}}^1; F^c)_i, E(\tilde{\mathbf{x}}^2; F^c)_i, \cdots, E(\tilde{\mathbf{x}}^{2n}; F^c)_i\right\}$. In this work, we define the decoy-enhanced saliency score Z_i for each feature $\mathbf{x}_i$ as

$$Z_i = \max(\tilde{E}_i) - \min(\tilde{E}_i) \,. \tag{3}$$

Here, Z_i is determined by the empirical range of the decoy saliency scores. Ideally, important features will have large values and unimportant ones will have small values. Note that the proposed method is designed specifically for nonlinear models in need of interpretation. As is discussed in Section A7, it exhibits ill-defined on linear models. It should also be noted that by sliding the swappable patch across the input and ensembling the obtained decoy-enhanced saliency maps, we could capture the saliency of each feature. The motivations of manipulating at a patch level rather than the entire input are capturing the local dependency structure and enabling batch operations for better efficiency.

3.5 THEORETICAL INSIGHTS

In this section, we analyze the saliency score method in a theoretical fashion. [3] In particular, we take a convolutional neural network with the ReLU activation function as an example to discuss why the proposed interpretation method can account for inter-feature dependence while also improving explanatory robustness. It should be noted that, while we conduct our theoretical analysis in the setting of CNNs with a specific activation function, the conclusions drawn from the theoretical analysis can easily be extended to other feed-forward neural architectures and other activation functions (*e.g.,* sigmoid and tanh). For analysis of other neural architectures, see Section A9.

Consider a CNN with L hidden blocks, with each layer ℓ containing a convolutional layer with a filter of size $\sqrt{s_\ell} \times \sqrt{s_\ell}$ and a max pooling layer with pooling size $\sqrt{s_\ell} \times \sqrt{s_\ell}$. (We set the pooling size the same as the kernel size in each block for simplicity.) The input to this CNN is $\mathbf{x} \in \mathbb{R}^d$, unrolled from a $\sqrt{d} \times \sqrt{d}$ matrix. Similarly, we also unroll each convolutional filter into $\mathbf{g}_\ell \in \mathbb{R}^{s_\ell}$, where $\mathbf{g}_\ell$ is indexed as $(\mathbf{g}_\ell)_j$ for $j \in \mathcal{J}_\ell$. Here, $\mathcal{J}_\ell$ corresponds to the index shift in matrix form from the top-left to bottom-right element. For example, a 3×3 convolutional filter (*i.e.,* $s_\ell = 9$) is indexed

[3] Note that we developed the theoretical properties by using the vanilla gradient as the base saliency method.

by $\mathcal{J}_\ell = \left\{-\sqrt{d}-1, -\sqrt{d}, -\sqrt{d}+1, -1, 0, 1, \sqrt{d}-1, \sqrt{d}, \sqrt{d}+1\right\}$. The output of the network is the probability vector $\mathbf{p} \in \mathbb{R}^C$ generated by the softmax function, where C is the total number of classes. Such a network can be represented as

$$\begin{aligned} &\mathbf{m}_\ell = \text{pool}(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})) \;\text{ for }\; \ell = 1, 2, 3, ..., L\,, \\ &\mathbf{o} = \mathbf{W}_{L+1}^T \mathbf{m}_L + \mathbf{b}_{L+1}, \\ &\mathbf{p} = \text{softmax}(\mathbf{o})\,, \end{aligned} \tag{4}$$

where relu$(\cdot)$ and pool$(\cdot)$ indicate the ReLU and pooling operators, $\mathbf{m}_\ell \in \mathbb{R}^{d_\ell}$ is the output of the block ℓ ($\mathbf{m}_0 = \mathbf{x}$), and $(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}) \in \mathbb{R}^{d_{\ell-1}}$ represents a convolutional operation on that block. We assume for simplicity that the convolution retains the input shape.

Consider an input $\mathbf{x}$ and its decoy $\tilde{\mathbf{x}}$, generated by swapping features in $\mathcal{K}$. For each feature $i \in \mathcal{K}$, we have the following theorem for the decoy-enhanced saliency score Z_i:

Theorem 1. In the aforementioned setting, Z_i is bounded by

$$\left| Z_i - \frac{1}{2} \left| \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_\mathbf{x})_{k,i} \right| \right| \leq C_1\,. \tag{5}$$

Here, $C_1 > 0$ is a bounded constant, and $\mathbf{H}_\mathbf{x}$ is the Hessian of $F^c(\mathbf{x})$ on $\mathbf{x}$ where $(\mathbf{H}_\mathbf{x})_{i,k} = \frac{\partial^2 F^c}{\partial \mathbf{x}_i \partial \mathbf{x}_k}$. $\tilde{\mathbf{x}}^+$ and $\tilde{\mathbf{x}}^-$ refer to the decoy that maximizes and minimizes $E(\tilde{\mathbf{x}}; F^c)$, respectively. See Section A7 for the proof. Theorem 1 implies that the proposed saliency score is determined by the second-order Hessian ($(\mathbf{H}_\mathbf{x})_{i,k}$) in the same swappable feature set. The score explicitly models the feature dependencies in the swappable feature set via this second-order Hessian, potentially capturing meaningful patterns such as edges, texture, etc.

In addition to enabling representation of inter-feature dependence, Theorem 1 sheds light on the robustness of the proposed saliency score against adversarial attack. To illustrate the robustness improvement of our method, we introduce the following proposition. The proof of this proposition as well as in-depth analysis can be found in Section A8.

Proposition 1. Given an input $\mathbf{x}$ and the corresponding adversarial sample $\hat{\mathbf{x}}$, if both $|\mathbf{x}_i - \tilde{\mathbf{x}}_i| \leq C_2\delta_i$ and $\left|\hat{\mathbf{x}}_i - \tilde{\hat{\mathbf{x}}}_i\right| \leq C_2\delta_i$ can obtain where $C_2 > 0$ is a bounded constant, then the following relation can be guaranteed.

$$|(Z_{\hat{\mathbf{x}}})_i - (Z_\mathbf{x})_i| \leq |E(\hat{\mathbf{x}}, F)_i - E(\mathbf{x}, F)_i|\,. \tag{6}$$

Given an adversarial sample $\hat{\mathbf{x}}$ (*i.e.,* the perturbed $\mathbf{x}$), we say a saliency method is not robust against $\hat{\mathbf{x}}$ if the deviation of the corresponding explanation $\delta_i = |E(\hat{\mathbf{x}}, F)_i - E(\mathbf{x}, F)_i|$ (for all $i \in \{1, 2, \cdots, d\}$) is large. According to the proposition above, we can easily discover that the deviation of our decoy-enhanced saliency score is always no larger than that of other saliency methods when a certain condition is satisfied. This indicates that, when the condition holds, our saliency method can guarantee a stronger resistance to the adversarial perturbation. To ensure the conditions $|\mathbf{x}_i - \tilde{\mathbf{x}}_i| \leq C_2\delta_i$ and $\left|\hat{\mathbf{x}}_i - \tilde{\hat{\mathbf{x}}}_i\right| \leq C_2\delta_i$ obtain, we can further introduce the corresponding condition as a constraint to Eqn. 2. In the following section, without further clarification, the saliency scores used in our evaluation are all derived with this constraint imposed.

4 EXPERIMENTS

To evaluate the effectiveness of our proposed method, we perform extensive experiments on deep learning models that target three tasks: image classification, sentiment analysis, and network intrusion detection. The performance of our approach is assessed both qualitatively and quantitatively. The results show that our proposed method identifies intuitively more coherent saliency maps than the state-of-the-art saliency methods alone. The method also achieves quantitatively better alignment to truly important features and demonstrates stronger robustness to adversarial manipulation. The description of the datasets and experimental setup can be found in Section A10.

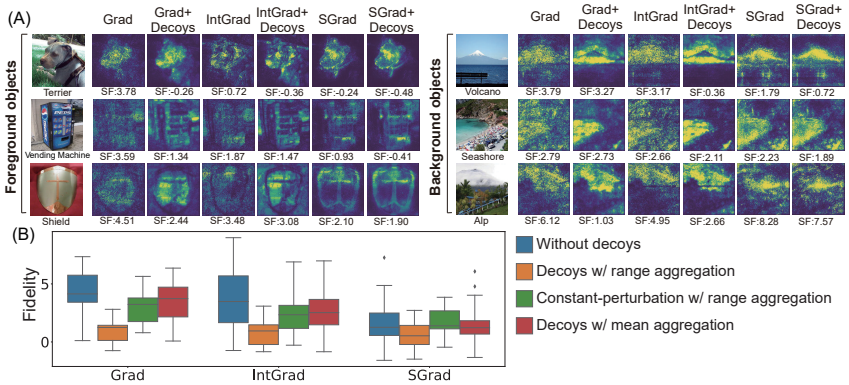


Figure 1: **Performance evaluation on ImageNet.** (A) Visualization of saliency maps on foreground and background objects. (B) Fidelity comparison of original saliencies (*i.e.,* "Without decoys"), our method (*i.e.,* "Decoys w/ range aggregation"), and its alternatives: replacing the decoy generation (Eqn. 2) with constant perturbation (*i.e.,* "Constant-perturbation w/ range aggregation"); replacing the decoy aggregation (Eqn. 3) with mean aggregation (*i.e.,* "Decoys w/ mean aggregation").

4.1 Saliency benchmark

As mentioned in Section 2, we apply our decoy enhancement method to three saliency methods: vanilla gradient, SmoothGrad, and integrated gradient. [4] In each case, the decoy-enhanced saliency scores are post-processed in the following way before qualitative and quantitative evaluations. First, we follow the existing methods (Simonyan et al., 2013) and compute the absolute saliency scores. For images, to obtain a single importance score for each pixel, we use the maximum absolute saliency score across all color channels. To avoid outlier features with extremely high saliency values leading to almost zero saliency scores for the other features, we then winsorized outlier saliency values to a relatively high value (the 95th percentile), as suggested by Smilkov et al. (2017) before linearly scaling to the range $[0, 1]$. To demonstrate that all three methods, when enhanced with decoys, still depend on the predictor, we carry out a sanity check on the ImageNet dataset. The results show that our decoy enhanced-saliency methods pass the sanity check (see Section A11 for details).

4.2 Performance in various applications

To comprehensively evaluate our proposed approach against the baselines mentioned above, we focus on two criteria. First, we aim to achieve qualitative coherence of the identified saliency map. Intuitively, we prefer a saliency method that highlights features that align closely with the predictions (*e.g.,* highlights the object of interest in an image or the words indicating the sentiment of the sentence). Second, to quantify the correctness of the saliency maps produced by the corresponding saliency method, we use the fidelity metric (Dabkowski & Gal, 2017), defined as $SF(E(\cdot; F^c), \mathbf{x}) = -\log \frac{F^c(E(\mathbf{x};F^c)\circ\mathbf{x})}{F^c(\overline{E(\mathbf{x};F^c)}\circ\mathbf{x})}$, where c indicates the predicted class of input $\mathbf{x}$, and $E(\mathbf{x}; F^c)$ is the normalized saliency map described above. $E(\mathbf{x}; F^c) \circ \mathbf{x}$ performs entry-wise multiplication between $E(\mathbf{x}; F^c)$ and $\mathbf{x}$, encoding the overlap between the object of interest and the concentration of the saliency map. The rationale behind this metric is as follows. By viewing the saliency score of the feature as its contribution to the predicted class, a good saliency method will weight important features more highly than less important ones and thus give rise to higher predicted class scores and lower metric values. Note that we subtract the mean saliency $\overline{E(\mathbf{x}; F^c)}$ to eliminate the influence of bias in $E(\mathbf{x}; F^c)$ and exclude trivial cases such as $E(\mathbf{x}; F^c) = \mathbf{1}$.

4.2.1 Performance on the ImageNet dataset

We applied our decoy-enhanced saliency score to randomly sampled images from the ImageNet dataset (Russakovsky et al., 2015), with a pretrained VGG16 model (Simonyan & Zisserman, 2014).

[4] Regarding the integrated gradient, we applied the default setting of a single zero baseline. Section A14 shows that our method can also improve the variants of the integrated gradient.

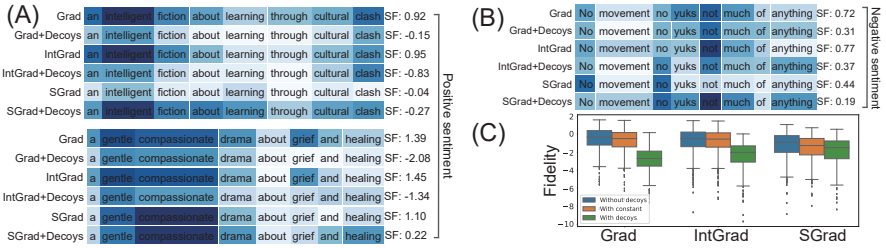


Figure 2: **Performance evaluation on the SST dataset.** (A) and (B) Visualization of saliency maps in each word. (C) The decoy-enhanced saliency score is compared against the original saliency score and using constant-approximated decoys, evaluated by fidelity. See A18 for more examples.

See Section A12 for applicability of our method to diverse CNN architectures such as AlexNet (Krizhevsky et al., 2012) and ResNet (He et al., 2016). The 3×3 image patches are treated as swappable features in generating decoys.

A side-by-side comparison (Figure 1(A)) suggests that decoys consistently help to reduce noise and produce more visually coherent saliency maps. For example, the original integrated gradient method highlights the region of dog head in a scattered format. In contrast, the decoy-enhanced integrated gradient method not only highlights the missing body but also identifies the dog head with more details such as ears, cheek, and nose (See Section A18 for more visualization examples). The visual coherence is also quantitatively supported by the saliency fidelity (Figure 1(B) blue and orange bars.).

To further evaluate the necessity of the two steps (*i.e.,* decoy generation and aggregation) in our method, we carried out a control experiment by replacing either step with alternatives. Specifically, as alternatives to the decoy generation, we used an image in which all pixel values are replaced with a single mean pixel value. Regarding the decoy aggregation, we calculated the mean saliency score as the alternative. As shown in Figure 1(B) above, our method, which incorporate both steps, reports the best performance. This validates the effectiveness of each of our designs.

4.2.2 Performance on the Stanford Sentiment Treebank (SST) dataset

We also applied our decoy-enhanced saliency score to randomly sampled sentences from the Stanford Sentiment Treebank (SST) (Russakovsky et al., 2015). We train a two-layer CNN (Kim, 2014) which takes the pretrained word embeddings as input (Pennington et al., 2014) (see A10 for experimental details). As suggested by Guan et al. (2019), the average saliency value of all dimensions of a word embedding is regarded as the word-level saliency value. The embeddings of the words are treated as swappable features when generating decoys.

As shown in Figure 2(A) and (B), a side-by-side comparison suggests that decoys consistently help to produce semantically more meaningful saliency maps. For example, in a sentence with negative sentiment, keywords associated with negation, such as 'no' and 'not', are more highlighted by decoy-enhanced saliency methods. The semantic coherence is also quantitatively supported by the saliency fidelity (Figure 2(C)). The constant-decoy (*i.e.,* the mean embedding of all the sentences) baseline also achieves worse fidelity than the decoys generated by Eqn. 2. Note that other decoy proxies (*e.g.,* random noise, blurring) are invalid for text data.

4.3 Robustness to adversarial attacks

Next we investigate the robustness of our method to adversarial manipulations of images.In particular, we focus on three popular adversarial attacks (Ghorbani et al., 2017): (1) the top-k attack, which seeks to decrease the scores of the top k most important features, (2) the target attack, which aims to increase the importance of a pre-specified region in the input image, and (3) the mass-center attack, which aims to spatially change the center of mass of the original saliency map. Here, we specify the bottom-right 4×4 region of the original image for the target attack and select $k = 5000$ in the top-k attack. We use the sensitivity metric (Alvarez-Melis & Jaakkola, 2018) to quantify the robustness of a saliency method E to adversarial attack, defined as $SS(E(\cdot, F^c), \mathbf{x}, \hat{\mathbf{x}}) = \frac{\|(E(\mathbf{x},F^c)-E(\hat{\mathbf{x}},F^c))\|_2}{\|\mathbf{x}-\hat{\mathbf{x}}\|_2}$,

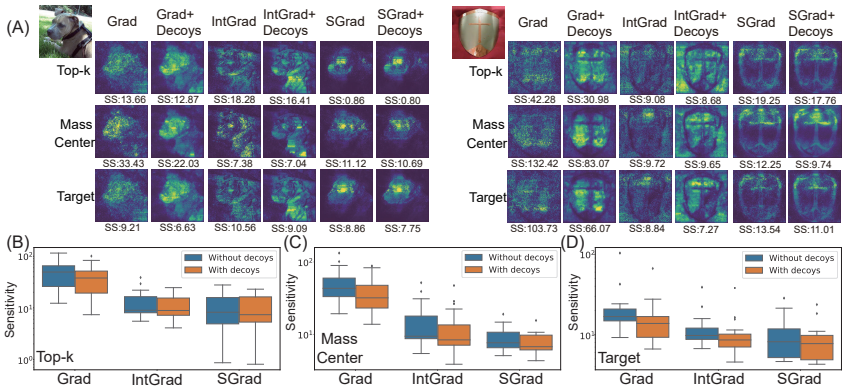


Figure 3: **Robustness to adversarial attacks on images.** (A) Visualization of saliency maps under adversarial attacks. (B)∼(D) The decoy-enhanced saliency score is compared to the original saliency score under adversarial attacks, evaluated by sensitivity.

where $\hat{\mathbf{x}}$ is the perturbed image of $\mathbf{x}$. A small sensitivity value means that similar inputs do not lead to substantially different saliency maps.

As shown in Figure 3(A), a side-by-side comparison suggests that decoys consistently yield low sensitivity scores and help to produce more visually coherent saliency maps, mitigating the impact of various adversarial attacks. More examples can be found in Section A18. The visual coherence and robustness to adversarial attacks are also quantitatively supported by Figure 3(B)∼(D). As is mentioned above, we also did experiments on a MLP trained with a network intrusion dataset and show the results in Section A13. The results are consistent with those on CNNs, which confirm our method's applicability to the widely-used feed-forward networks.

5 Discussion and Conclusion

In this work, we propose a method for computing, from a given saliency method, decoy-enhanced saliency scores that yield more accurate and robust saliency maps. We formulate the decoy generation as an optimization problem, applicable to diverse deep neural network architecture. We demonstrate the superior performance of our method relative to three standard saliency methods, both qualitatively and quantitatively, even in the presence of various adversarial perturbations to the image. From a theoretical perspective, by deriving a closed-form solution, we show that the proposed score can provably compensate for the limitations of existing saliency methods by reflecting the joint effects from other dependent features and maintaining robustness to adversarial perturbations.

Although decoy generation introduces extra computational overhead on top of existing saliency methods, the run time of generating one decoy is faster than existing saliency methods (See Section A15). Since decoy generation can run parallelly in a batch mode, the extra overhead does not jeopardize our method's application to large datasets. Our method has three hyperparameters: swappable feature size K, network layer ℓ, and initial Lagrange multiplier λ. In Section A16, we show that our method is insensitive to the substantial variation of hyperparameters. We generate decoys by using Eqn. 2. While there are other widely used perturbation methods (*e.g.,* random noise, blurring, and inpainting), they are not suitable for generating decoys. First, without ensuring the swappable condition in Eqn. 1, they cannot provide a theoretical guarantee for robustness improvement. Second, methods like blurring and inpainting are not well-defined for applications beyond computer vision.

This work points to several promising directions for future research. First, a possible extension is to customize our method to recurrent neural networks and to inputs with categorical/discrete features. Second, recent work (Bansal et al., 2020; Chen et al., 2019c) shows that adversarial training can improve the interpretability of a DNN model. It is worth exploring whether our method could further enhance the quality of saliency maps derived from these adversarially retrained classifiers. A third promising direction could be reframing interpretability as hypothesis testing and using decoys to deliver a set of salient features, subject to false discovery rate control at some pre-specified level (Burns et al., 2019; Lu et al., 2018).

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. of NeurIPS*, 2018.

David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Proc. of NeurIPS*, 2018.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *Proc. of ICLR*, 2018.

Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. *arXiv preprint arXiv:2003.08754*, 2020.

Rina Foygel Barber and Emmanuel J Candès. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 2015.

Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In *Proc. of ICANN*, 2016.

Collin Burns, Jesse Thomason, and Wesley Tansey. Interpreting black box models via hypothesis testing. *arXiv:1904.00045*, 2019.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of S&P*, 2017.

Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. PCANet: A simple deep learning baseline for image classification. *IEEE Transactions on Image Processing*, 2015.

Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. In *Proc. of ICLR*, 2019.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Proc. of NeurIPS*, 2019a.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *Proc. of ICML*, 2018.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. In *Proc. of ICLR*, 2019b.

Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. Robust attribution regularization. In *Proc. of NeurIPS*, 2019c.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proc. of NeurIPS*, 2017.

Lijie Fan, Shengjia Zhao, and Stefano Ermon. Adversarial localization network. In *Proc. of NeurIPS LLD Workshop*, 2017.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. of ICCV*, 2017.

Canadian Institute for Cybersecurity. Cse-cic-ids2018 on aws. `https://www.unb.ca/cic/datasets/ids-2018.html`, 2018.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. *arXiv:1710.10547*, 2017.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Proc. of NeurIPS*, 2019.

Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. Counterfactual visual explanations. *Proc. of ICML*, 2019.

Chaoyu Guan, Xiting Wang, Quanshi Zhang, Runjin Chen, Di He, and Xing Xie. Towards a deep and unified understanding of deep neural models in nlp. In *Proc. of ICML*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. of ICLR*, 2019.

Kouichi Ikeno and Satoshi Hara. Maximizing invariant data perturbation with stochastic optimization. *arXiv preprint arXiv:1807.05077*, 2018.

Yoon Kim. Convolutional neural networks for sentence classification. *Proc. of EMNLP*, 2014.

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:1711.00867*, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *Proc. of ICML*, 2017.

Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. In *Proc. of NeurIPS*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. of NeurIPS*, 2012.

Alexander Levine, Sahil Singla, and Soheil Feizi. Certifiably robust interpretation in deep learning. *arXiv preprint arXiv:1905.12105*, 2019.

Zachary C Lipton. The mythos of model interpretability. *arXiv:1606.03490*, 2016.

Yang Lu, Yingying Fan, Jinchi Lv, and William Stafford Noble. DeepPINK: reproducible feature selection in deep neural networks. In *Proc. of NeurIPS*, 2018.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. of NeurIPS*, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proc. of CVPR*, 2017.

Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *Proc. of ICML*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proc. of EMNLP*, 2014.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. of KDD*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *Proc. of ICML*, 2011.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. 2020.

Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-CAM: Why did you say that? *arXiv:1611.07450*, 2016.

Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Prof. of ICISSP*, 2018.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proc. of ICML*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*, 2013.

Sahil Singla, Eric Wallace, Shi Feng, and Soheil Feizi. Understanding impacts of high-order loss approximations and features in deep learning interpretation. *arXiv:1902.00407*, 2019.

Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. In *Proc. of ICML*, 2020.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv:1706.03825*, 2017.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proc. of ICML*, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

Mariya Toneva and Leila Wehbe. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). In *Proc. of NeurIPS*, 2019.

Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. In *Proc. of NeurIPS*, 2018.

Roozbeh Yousefzadeh and Dianne P O'Leary. Interpreting neural networks using flip points. *arXiv preprint arXiv:1903.08789*, 2019.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. of ECCV*, 2014.

Konrad Zołna, Krzysztof J Geras, and Kyunghyun Cho. Classifier-agnostic saliency map extraction. In *Proceedings of AAAI*, 2019.

A6 IMPLEMENTATION DETAILS

The optimization function proposed to generate decoys is non-differentiable and very difficult to solve; hence, we instead solve an alternate formulation with the help of the following tricks. First, we introduce a Lagrange multiplier $\lambda > 0$ and augment the first constraint in the optimization function as a penalty in the objective function. This will rule out the hyper-parameter ϵ in Eqn. 2. Second, we use projected gradient descent during the optimization to eliminate the mask constraint (*i.e.,* $(\tilde{\mathbf{x}} - \mathbf{x}) \circ (1 - \mathcal{M}) = 0$). Specifically, after each standard gradient descent step, we enforce $\tilde{\mathbf{x}} = \tilde{\mathbf{x}} \circ \mathcal{M} + \mathbf{x} \circ (1 - \mathcal{M})$. Third, we use the change-of-variable trick (Carlini & Wagner, 2017) to eliminate the feature value constraint (*i.e.,* $\tilde{\mathbf{x}} \in [\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{max}}]^d$). Instead of directly optimizing $\tilde{\mathbf{x}}$, we first normalize it to $[0, 1]$ and introduce $\hat{\mathbf{x}}$ satisfying $\tilde{\mathbf{x}}_i = \frac{1}{2}(\tanh(\hat{\mathbf{x}}_i) + 1)$, for all $i \in \{1, 2, \cdots, d\}$. Because $\tanh(\hat{\mathbf{x}}_i) \in [-1, 1]$ implies $\tilde{\mathbf{x}}_i \in [0, 1]$, any solution to $\hat{\mathbf{x}}$ is naturally valid. It should be noted that other transformations for this third step are also possible but were not explored in this paper. Putting these ideas together, we minimize the following objective function:

$$\text{minimize}_{\hat{\mathbf{x}}} \; -\left\| (\frac{1}{2}(\tanh(\hat{\mathbf{x}}) + 1) - \mathbf{x}) \cdot s)^+ \right\|_1 + \lambda \cdot \left\| F_\ell(\frac{1}{2}(\tanh(\hat{\mathbf{x}}) + 1)) - F_\ell(\mathbf{x}) \right\|_\infty , \quad (7)$$

where $\lambda > 0$ is initialized small and repeatedly doubled until the optimization succeeds. Because the L_∞ norm is not fully differentiable, we adopt the approximation trick introduced by Carlini & Wagner (2017) and solve the following formulation:

$$\text{minimize}_{\hat{\mathbf{x}}} \; -\left\| \max((\frac{1}{2}(\tanh(\hat{\mathbf{x}})+1)-\mathbf{x})\cdot s, 0) \right\|_1 + \lambda \cdot \left\| (|F_\ell(\frac{1}{2}(\tanh(\hat{\mathbf{x}})+1)) - F_\ell(\mathbf{x})| - \tau)^+ \right\|_2^2 , \tag{8}$$

where $\tau > 0$. In this paper, we follow the selection strategy proposed in Carlini & Wagner (2017) and initialize $\tau = 1$. After each iteration, if the second term is zero, then we reduce τ by a factor of 0.95 and repeat; otherwise, we terminate the optimization. After obtaining $\hat{\mathbf{x}}$, we compute $\tilde{\mathbf{x}}$ and map it back to the original feature value range $[\mathbf{x}_{\text{min}}, \mathbf{x}_{\text{max}}]$. Note that Eqn. 8 can be efficiently solved by any first-order optimization method without introducing too much computational overhead. In practice, the average run time of solving it is 62.3% shorter than the fastest, vanilla gradient method.

A7 Proof of Theorem 1

Before proving Theorem 1, we first state and prove the following lemma.

Lemma 1. *Consider an input $\mathbf{x}$ and its decoy $\tilde{\mathbf{x}}$, generated by replacing the original features with swappable features in $\mathcal{K}$, $|\mathcal{K}| = K$. The partial derivative of $F^c(\tilde{\mathbf{x}})$ w.r.t. to $\tilde{\mathbf{x}}_i$ for $i \in \mathcal{K}$ is*

$$\left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i - \frac{1}{2}\sum_{k\in\mathcal{K}} (\tilde{\mathbf{x}}_k - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| \leq C \,. \tag{9}$$

Proof. The second-order Taylor expansion of the predicted $F^c(\mathbf{x})$ for target class c around $\mathbf{x}$ is as follows:

$$F^c(\mathbf{x}) \approx F^c(\tilde{\mathbf{x}}) + \nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}})^T \Delta + \frac{1}{2}\Delta^T \mathbf{H}_{\tilde{\mathbf{x}}} \Delta \,, \tag{10}$$

where $\Delta = \mathbf{x} - \tilde{\mathbf{x}}$. By definition of the decoys in Section 2.2 (*i.e.,* $F^c(\mathbf{x}) = F^c(\tilde{\mathbf{x}})$), the following equation holds:

$$\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}})^T \Delta \approx -\frac{1}{2}\Delta^T \mathbf{H}_{\tilde{\mathbf{x}}} \Delta \,. \tag{11}$$

From the above equation, we can see that, for a linear model, the linearity zeroes out the gradient of the decoys, causing the ill-defined behavior of our method. This does not dilute the significance of our method because linear models are self-explainable and do not need extra post hoc explanation tools. We clarified this in Section 3.4 that our method is mainly defined for non-linear models since non-linear models are those most in need of interpretation.

Given a swappable patch of size $K \times 1$ starting from position i_1, then $\Delta = [0, ..., \mathbf{x}_{i_1} - \tilde{\mathbf{x}}_{i_1}, ..., \mathbf{x}_{i_K} - \tilde{\mathbf{x}}_{i_K}, 0, ..., 0]$. As such, we have

$$\begin{aligned} \nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}})^T \Delta &= \sum_{i\in\mathcal{K}} (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i (\mathbf{x}_i - \tilde{\mathbf{x}}_i) \,, \\ \Delta^T \mathbf{H}_{\tilde{\mathbf{x}}} \Delta &= \sum_{i\in\mathcal{K}} (\mathbf{x}_i - \tilde{\mathbf{x}}_i) \sum_{k\in\mathcal{K}} (\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} (\mathbf{x}_k - \tilde{\mathbf{x}}_k) \,. \end{aligned} \tag{12}$$

Plugging Eqn. (12) into Eqn. (11), we have

$$\sum_{i\in\mathcal{K}} [(\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i + \frac{1}{2}\sum_{k\in\mathcal{K}} (\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} (\mathbf{x}_k - \tilde{\mathbf{x}}_k)](\mathbf{x}_i - \tilde{\mathbf{x}}_i) = 0 \,. \tag{13}$$

Then we can derive

$$\begin{aligned} &\left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i + \frac{1}{2}\sum_{k\in\mathcal{K}} (\mathbf{x}_k - \tilde{\mathbf{x}}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| \leq C \,, \\ &\left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i - \frac{1}{2}\sum_{k\in\mathcal{K}} (\tilde{\mathbf{x}}_k - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| \leq C \,. \end{aligned} \tag{14}$$

First, we can derive $|\tilde{\mathbf{x}}_i - \mathbf{x}_i|$ is bounded by $2\max(\mathbf{x}_{\max}, |\mathbf{x}_{\min}|)$. We also have $|\tilde{\mathbf{x}}_{i+k} - \mathbf{x}_{i+k}|\ 0$ in that we can always find a small perturbation to each feature in $\mathbf{x}$ such that $\|F_\ell(\tilde{\mathbf{x}}) - F_\ell(\mathbf{x})\|_\infty \leq \epsilon$. In addition, both gradient and Hessian are bounded by some Lipschitz constant (Szegedy et al., 2013). [5] As a result, we can always find a constant C, such that $C \geq \left| \frac{-\sum_{k_1 \in \mathcal{K}\setminus i}[(\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_{k_1} + \frac{1}{2}\sum_{k_2 \in \mathcal{K}} (H_{\tilde{\mathbf{x}}})_{k_1,k_2}(\mathbf{x}_{k_2} - \tilde{\mathbf{x}}_{k_2})](\mathbf{x}_{k_1} - \tilde{\mathbf{x}}_{k_1})}{(\mathbf{x}_i - \tilde{\mathbf{x}}_i)} \right|$. For the case $K = 1$, we have $(\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i = \frac{1}{2}(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,i}(\tilde{\mathbf{x}}_i - \mathbf{x}_i)$.

□

Now we prove Theorem 1 from Section 3.5.

Consider a CNN with L hidden blocks, with each layer ℓ containing a convolutional layer with a filter of size $\sqrt{s_\ell} \times \sqrt{s_\ell}$ and a max pooling layer with pooling size $\sqrt{s_\ell} \times \sqrt{s_\ell}$. The input to this CNN is $\mathbf{x} \in \mathbb{R}^d$, unrolled from a $\sqrt{d} \times \sqrt{d}$ matrix. Similarly, we also unroll each convolutional filter into $\mathbf{g}_\ell \in \mathbb{R}^{s_\ell}$, where $\mathbf{g}_\ell$ is indexed as $(\mathbf{g}_\ell)_j$ for $j \in \mathcal{J}_\ell$. Here, $\mathcal{J}_\ell$ corresponds to the index shift in matrix form from the top-left to bottom-right element. The output of the network is the probability vector $\mathbf{p} \in \mathbb{R}^C$ generated by the softmax function, where C is the total number of classes. Such a network can be represented as

$$\begin{aligned} &\mathbf{m}_\ell = \text{pool}(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})) \ \text{ for } \ \ell = 1, 2, 3, ..., L\,, \\ &\mathbf{o} = \mathbf{W}_{L+1}^T \mathbf{m}_L + \mathbf{b}_{L+1}, \\ &\mathbf{p} = \text{softmax}(\mathbf{o})\,, \end{aligned} \tag{15}$$

where relu$(\cdot)$ and pool$(\cdot)$ indicate the ReLU and pooling operators, $\mathbf{m}_\ell \in \mathbb{R}^{d_\ell}$ is the output of the block ℓ ($\mathbf{m}_0 = \mathbf{x}$), and $(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}) \in \mathbb{R}^{d_{\ell-1}}$ represents a convolutional operation on that block.

Consider an input $\mathbf{x}$ and its decoy $\tilde{\mathbf{x}}$, generated by swapping features in $\mathcal{K}$. For each feature $i \in \mathcal{K}$, we have the following theorem for the decoy-enhanced saliency score Z_i:

Theorem 1. *In the aforementioned setting, Z_i is bounded by*

$$\left| Z_i - \frac{1}{2} \left| \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{k,i} \right| \right| \leq C_1\,. \tag{16}$$

Proof. The gradient of $\mathbf{p}_c$ with respect to $\mathbf{x}$ can be written as follows, using the denominator layout notation of the derivative of a vector:

$$\nabla_{\mathbf{x}} \mathbf{p}_c = \prod_{\ell=1}^{L} \frac{\partial \mathbf{m}_\ell}{\partial \mathbf{m}_{\ell-1}} \frac{\partial \mathbf{o}}{\partial \mathbf{m}_L} \frac{\partial \mathbf{p}_c}{\partial \mathbf{o}}, \tag{17}$$

where

$$\frac{\partial \mathbf{o}}{\partial \mathbf{m}_L} = \mathbf{W}_{L+1}\,, \tag{18}$$

and

$$\begin{cases} \frac{\partial \mathbf{p}_c}{\partial \mathbf{o}_{c'}} = (\mathbf{p}_c - \mathbf{p}_c^2) & \text{if } c' = c\,, \\ \frac{\partial \mathbf{p}_c}{\partial \mathbf{o}_{c'}} = -\mathbf{p}_c \mathbf{p}_{c'} & \text{otherwise}\,. \end{cases} \tag{19}$$

Then we can write $\frac{\partial \mathbf{p}_c}{\partial \mathbf{o}}$ as follows:

$$\frac{\partial \mathbf{p}_c}{\partial \mathbf{o}} = \hat{\mathbf{P}}_{\cdot c}\,, \tag{20}$$

where $\hat{\mathbf{P}}_{\cdot c}$ corresponds to the c-th column of $\hat{\mathbf{P}}$ and $\hat{\mathbf{P}} = \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$. We then define $\mathbf{B}_\ell = \frac{\partial \mathbf{m}_\ell}{\partial \mathbf{m}_{\ell-1}}$ as $\mathbf{B}_\ell \in \mathbb{R}^{d_{\ell-1} \times d_\ell}$. In the following, we compute $\mathbf{B}_\ell$.

First, we can have

$$\begin{cases} \frac{\partial (\mathbf{m}_\ell)_j}{\partial (\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n} = 1 & \text{if } \hat{j} - n \in \mathcal{J}_\ell, \text{ and } n = \text{argmax}_{n' \in \hat{j} + \mathcal{J}_\ell} (\mathbf{g}_\ell * \mathbf{m}_{\ell-1})_{n'}\,, \\ \frac{\partial (\mathbf{m}_\ell)_j}{\partial (\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n} = 0 & \text{otherwise}\,, \end{cases} \tag{21}$$

[5] Following other works that also utilized Lipschitz continuity to analyze DNNs (Szegedy et al., 2013; Ghorbani et al., 2017), we assume that F_ℓ is locally continuous around $\mathbf{x}$, for $\ell = 1, 2, ..., L$.

where $\hat{j}$ represents the center of the pooling patch in $\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})$, which results in $(\mathbf{m}_\ell)_j$. Then we can compute

$$\begin{cases} \frac{\partial(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n}{\partial(\mathbf{m}_{\ell-1})_i} = (a_\ell)_n(\mathbf{g}_\ell)_{n-i} \quad \text{if } n-i \in \mathcal{J}_\ell\,, \\ \frac{\partial(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n}{\partial(\mathbf{m}_{\ell-1})_i} = 0 \quad \text{otherwise}\,, \end{cases} \tag{22}$$

where $(a_\ell)_n = \mathbf{1}\left\{(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})_n) \geq 0\right\}$. If we change the activation function to either sigmoid or tanh, then $(a_\ell)_n$ in Eqn. 22 will be replaced with the derivative of either function. For the sigmoid activation function $\sigma(x)$, the derivative is $\sigma(x)(1-\sigma(x))$, with a range of $[0, \frac{1}{4}]$. For the tanh activation function $tanh(x)$, the derivative is $1 - tanh(x)^2$, with a range of $[0, 1]$. We conclude that the derivative of both sigmoid and tanh are bounded by a value no larger than 1.

Combining Eqn. 21 with 22, we have

$$\begin{cases} (\mathbf{B}_\ell)_{ij} = \frac{\partial(\mathbf{m}_\ell)_j}{\partial(\mathbf{m}_{\ell-1})_i} = (a_\ell)_n(\mathbf{g}_\ell)_{n-i} \quad \text{if } n-i \in \mathcal{J}_\ell,\ \hat{j}-n \in \mathcal{J}_\ell,\ \text{and } n = \text{argmax}_{n' \in \hat{j}+\mathcal{J}_\ell}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})_{n'}\,, \\ (\mathbf{B}_\ell)_{ij} = \frac{\partial(\mathbf{m}_\ell)_j}{\partial(\mathbf{m}_{\ell-1})_i} = 0 \quad \text{otherwise}\,. \end{cases} \tag{23}$$

For simplicity, we rewrite the non-zero condition as $n \in \hat{\mathcal{J}}_\ell$. Plugging $\mathbf{B}_\ell$, $\ell = 1, ..., L$, into Eqn. 17, we can obtain the partial derivative $\nabla_{\mathbf{x}}\mathbf{p}_c$.

Further, we compute each element in the Hessian matrix $\mathbf{H}_{ij}$ as follows:

$$\begin{aligned} \mathbf{H}_{ij} = \nabla_{\mathbf{x}_i}(\nabla_{\mathbf{x}_j}\mathbf{p}_c) &= \frac{\partial(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{j\cdot}\mathbf{W}_{L+1}\hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} \\ &= (\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{j\cdot}\mathbf{W}_{L+1}\frac{\partial\hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} = \left(\sum_{n_L=1}^{d_L}(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L}(\mathbf{W}_{L+1})_{n_L\cdot}\right)\frac{\partial\hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}\,, \end{aligned} \tag{24}$$

and

$$\frac{\partial\hat{\mathbf{P}}_{c'c}}{\partial \mathbf{x}_i} = \begin{cases} (1-2\mathbf{p}_c)\nabla_{\mathbf{x}_i}\mathbf{p}_c \quad \text{if } c' = c\,, \\ \mathbf{p}_c\nabla_{\mathbf{x}_i}\mathbf{p}_{c'} + \mathbf{p}_{c'}\nabla_{\mathbf{x}_i}\mathbf{p}_c \quad \text{otherwise}\,. \end{cases} \tag{25}$$

Now we compute $(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L}$ as

$$(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell(\mathbf{B}_L)_{\cdot n_L}\,, \tag{26}$$

where

$$(\mathbf{B}_1)_{j\cdot}\mathbf{B}_2 = [0, ..., C_{n_2}(a_2)_{n_2}\sum_{n_1\in\hat{\mathcal{J}}_1}(a_1)_n\mathbf{g}_{n-1}, ..., 0]\,, \tag{27}$$

and where $C_{n_2} = (g_2)_{n_2-2}\sum_{n_1\in\hat{\mathcal{J}}_1}\mathbf{g}_{n-1}$. Here, we redefine $\hat{\mathcal{J}}_1$ as the set of indices such that $(\mathbf{B}_1)_{jn_1} \neq 0$ for $n_1 \in \hat{\mathcal{J}}_1$. As such, we can compute $(\mathbf{B}_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell$ as

$$(\mathbf{B}_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell = [0, .., C_{n_{L-1}}(a_{L-1})_{n_{L-1}}\sum_{\ell=1}^{L-2}\sum_{n_\ell\in\hat{\mathcal{J}}_\ell}(a_\ell)_{n_\ell}, ..., 0]\,. \tag{28}$$

Plugging Eqn. 28 into Eqn. 26, we have

$$(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell(\mathbf{B}_L)_{\cdot n_L} = (C_L)_{n_L}(a_L)_{n_L}\sum_{\ell=1}^{L-1}\sum_{n_\ell\in\hat{\mathcal{J}}_\ell}(a_\ell)_{n_\ell}\,. \tag{29}$$

Plugging Eqn. 29 into Eqn. 24, we have

$$\mathbf{H}_{ij} = \left(C_j\sum_{\ell=1}^{L}\sum_{n_\ell\in\hat{\mathcal{J}}_\ell}(a_\ell)_{n_\ell}\right)\frac{\partial\hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}\,, \tag{30}$$

where C_j is a linear combination of $\mathbf{g}_1$, ..., $\mathbf{g}_L$, $\mathbf{W}_{L+1}$, which is bounded. $\mathbf{H}_{ij}$ equals the multiplication of two components—the summation of neurons activated by $\mathbf{x}$ and a gradient

$\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$. Given the total number of neurons in a CNN is a constant (denoted by C_T), we have $0 \leq \left(\sum_{\ell=1}^{L} \sum_{n_\ell \in \hat{\mathcal{J}}_\ell} (a_\ell)_{n_\ell}\right) \leq C_T$. Then, we have $|(\mathbf{H}_{\mathbf{x}})_{ij}| \leq C_T |C_j \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}|$. Since the derivatives of both sigmoid and tanh are no larger than 1, this inequality also applies to the network with these two functions as the activation function. Similarly, for the Hessian $(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij}$ of a decoy $\tilde{\mathbf{x}}$, we also have $|(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij} \leq C_T |C_j \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \tilde{\mathbf{x}}_i}|$. Given the inequality of $(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij}$ and $(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij}$, we can obtain that $|(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij} - (\mathbf{H}_{\mathbf{x}})_{ij}| \leq 2C_T \max(|\tilde{C}_j \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \tilde{\mathbf{x}}_i}|, |C_j \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i})|$, where $\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$ is given by Eqn. 25. Recalling that $\mathbf{P}_c$ is within $[0, 1]$, the gradient $\frac{\partial \mathbf{P}_c}{\partial \mathbf{x}_i}$ is bounded by some Lipschitz constant (Szegedy et al., 2013), we can obtain that $\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$ is bounded by some constant. Finally, we can derive that $|(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij} - (\mathbf{H}_{\mathbf{x}})_{ij}| \leq C_C$, where C_C represents the upper bound.[6]

Now, we derive the decoy-enhanced saliency score Z_i for $\mathbf{x}_i$, given a population of saliency scores $\tilde{E}_i = \left\{E(\tilde{\mathbf{x}}^1; F)_i, E(\tilde{\mathbf{x}}^2; F)_i, \cdots, E(\tilde{\mathbf{x}}^{2n}; F)_i\right\}$. Let $\tilde{\mathbf{x}}^+, \tilde{\mathbf{x}}^- \in \left\{\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \cdots, \tilde{\mathbf{x}}^{2n}\right\}$ denotes the decoy which maximizes and minimize $E(\tilde{\mathbf{x}}; F)_i$, respectively. According to Lemma 1, the partial derivative $\nabla_{\tilde{\mathbf{x}}_i} \mathbf{p}_c$ has the following relationship

$$\left|(\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k}\right| \leq C\,, \tag{31}$$

Then, we can derive

$$\frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} - C \leq (\nabla_{\tilde{\mathbf{x}}^+} F^c(\tilde{\mathbf{x}}^+))_i \leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} + C\,, \tag{32}$$

$$-\frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} - C \leq -(\nabla_{\tilde{\mathbf{x}}^-} F^c(\tilde{\mathbf{x}}^-))_i \leq -\frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} + C\,, \tag{33}$$

Then, we have

$$\begin{aligned}
Z_i &= (\nabla_{\tilde{\mathbf{x}}^+} F^c(\tilde{\mathbf{x}}^+))_i - (\nabla_{\tilde{\mathbf{x}}^-} F^c(\tilde{\mathbf{x}}^-))_i \\
&\leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} + 2C \\
&\leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)((\mathbf{H}_{\mathbf{x}})_{i,k} + C_C) - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)((\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} - C_C) + 2C \\
&\leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{i,k} + \frac{1}{2} C_C \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) + 2C\,,
\end{aligned} \tag{34}$$

And

$$\begin{aligned}
Z_i &= (\nabla_{\tilde{\mathbf{x}}^+} F^c(\tilde{\mathbf{x}}^+))_i - (\nabla_{\tilde{\mathbf{x}}^-} F^c(\tilde{\mathbf{x}}^-))_i \\
&\geq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} - 2C \\
&\geq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)((\mathbf{H}_{\mathbf{x}})_{i,k} - C_C) - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)((\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} + C_C) + 2C \\
&\geq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{i,k} - \frac{1}{2} C_C \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) - 2C\,,
\end{aligned} \tag{35}$$

Combining Eqn. 34 with Eqn. 35, we have

$$\left|Z_i - \frac{1}{2} \left|\sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{k,i}\right|\right| \leq C_1\,. \tag{36}$$

Recall that $(\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)$ is bounded by a upper-bound, we can obtain that there exist a constant C_1, such that $C_1 \geq \frac{1}{2} C_C \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) + 2C$.

□

[6]Note that this inequality cannot be directly obtained by the Lipschitz inequality, because the gradient may not be continuous.

A8 PROOF OF PROPOSITION 1

Proposition 1. *Given an input $\mathbf{x}$ and its corresponding adversarial sample $\hat{\mathbf{x}}$, if both $|\mathbf{x}_i - \tilde{\mathbf{x}}_i| \leq C_2\delta_i$ and $\left|\hat{\mathbf{x}}_i - \tilde{\hat{\mathbf{x}}}_i\right| \leq C_2\delta_i$ can obtain (where $C_2 > 0$ is a bounded constant), then the following relation can be guaranteed.*

$$|(Z_{\hat{\mathbf{x}}})_i - (Z_{\mathbf{x}})_i| \leq |(E(\hat{\mathbf{x}}, F)_i - E(\mathbf{x}, F))_i| \,. \tag{37}$$

Proof. Recall the goal of the attack against saliency maps is to subtly perturb an input sample such that the added perturbation does not change the output of the classifier (Ghorbani et al., 2017) but force a saliency method to output a less meaningful saliency map (*i.e.,* highlighting features that are irrelevant to the classifier prediction). To achieve this goal, when generating an adversarial sample $\hat{\mathbf{x}}$ from the given input $\mathbf{x}$, an attacker needs to impose the following constraint $\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty \leq \epsilon$. Suppose we have an adversarial sample $\hat{\mathbf{x}}$ satisfies this constraint. Then, we can assume $(\hat{\mathbf{x}} - \mathbf{x})_i = \hat{\boldsymbol{\epsilon}}_i$, where $|\hat{\boldsymbol{\epsilon}}_i| \leq \epsilon$, for $i = 1, 2, ..., d$. In addition, we can compute saliency maps $E(\hat{\mathbf{x}}, F)$ and $E(\mathbf{x}, F)$ for $\hat{\mathbf{x}}$ and $\mathbf{x}$ by using an existing saliency method. [7] Given both saliency maps, we can further compute the difference between $E(\hat{\mathbf{x}}, F)$ and $E(\mathbf{x}, F)$ as

$$(E(\hat{\mathbf{x}}, F) - E(\mathbf{x}, F))_i = \triangledown_{\hat{\mathbf{x}}} F^c(\hat{\mathbf{x}}) - \triangledown_{\mathbf{x}} F^c(\mathbf{x}) = (\mathbf{H}_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}))_i = \sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \hat{\boldsymbol{\epsilon}}_j \,. \tag{38}$$

Based on the Eqn. 2 in Section 3.3, when generating the decoys $\tilde{\mathbf{x}}$, we ensure the classifier's predictions for those decoys are as same as that of the $\mathbf{x}$. In this work, we achieve this by bounding the difference between the hidden representations of $\tilde{\mathbf{x}}$ and $\mathbf{x}$. As is discussed in Section A7, to preserve the same prediction c for $\tilde{\mathbf{x}}$ and $\mathbf{x}$, one has to ensure $|F^c(\tilde{\mathbf{x}}) - F^c(\mathbf{x})|$ is bounded. This implies the difference between $\tilde{\mathbf{x}}$ and $\mathbf{x}$ is bounded within $\boldsymbol{\epsilon}$. Here, $\boldsymbol{\epsilon}_i$ represents the maximum difference between $\tilde{\mathbf{x}}_i$ and $\mathbf{x}_i$ at the ith dimension. As is mentioned above, the adversarial sample $\hat{\mathbf{x}}$ does not change the classifier's prediction. Therefore, we could imply $\hat{\boldsymbol{\epsilon}}_i \leq \boldsymbol{\epsilon}_i$, for $i = 1, 2, ..., d$.

Now, suppose we obtain a set of decoys for $\mathbf{x}$ and have their corresponding saliency maps, i.e., $\left\{E(\tilde{\mathbf{x}}^1; F)_i, E(\tilde{\mathbf{x}}^2; F)_i, \cdots, E(\tilde{\mathbf{x}}^{2n}; F)_i)\right\}$. Let $\tilde{\mathbf{x}}^+ \in \left\{\tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \cdots, \tilde{\mathbf{x}}^n\right\}$ denote the decoys which maximize $E(\tilde{\mathbf{x}}; F)_i$ and let $\tilde{\mathbf{x}}^-$ denote the decoys which minimize $E(\tilde{\mathbf{x}}; F)_i$. Similarly, we can also have the corresponding decoys $\tilde{\hat{\mathbf{x}}}^-$ and $\tilde{\hat{\mathbf{x}}}^-$ for the adversarial sample $\hat{\mathbf{x}}$ as well as their corresponding saliency maps. With both the decoys and saliency maps for the input sample $\mathbf{x}$ and its adversarial sample $\hat{\mathbf{x}}$, we can compute the difference between $(Z_{\hat{\mathbf{x}}})_i$ and $(Z_{\mathbf{x}})_i$ as

$$\begin{aligned}
&(Z_{\hat{\mathbf{x}}})_i - (Z_{\mathbf{x}})_i \\
=&\left(E(\tilde{\hat{\mathbf{x}}}^+, F)_i - E(\tilde{\hat{\mathbf{x}}}^-, F)_i\right) - \left(E(\tilde{\mathbf{x}}^+, F)_i - E(\tilde{\mathbf{x}}^-, F)_i)\right) \\
=&\left((\mathbf{H}_{\mathbf{x}}(\tilde{\hat{\mathbf{x}}}^+ - \mathbf{x}))_i - (\mathbf{H}_{\mathbf{x}}(\tilde{\hat{\mathbf{x}}}^- - \mathbf{x}))_i\right) - \left((\mathbf{H}_{\mathbf{x}}(\tilde{\mathbf{x}}^+ - \mathbf{x}))_i - (\mathbf{H}_{\mathbf{x}}(\tilde{\mathbf{x}}^- - \mathbf{x}))_i\right) \\
=&\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right) \,.
\end{aligned} \tag{39}$$

To guarantee an improvement in robustness against the adversarial perturbation, we have to ensure that $|(Z_{\hat{\mathbf{x}}})_i - (Z_{\mathbf{x}})_i| - |(E(\hat{\mathbf{x}}, F) - E(\mathbf{x}, F))_i| \leq 0$, for $i = 1, 2..., d$. That is,

$$\begin{aligned}
&\left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right)\right| - \left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \hat{\boldsymbol{\epsilon}}_j\right| \leq 0, \\
&\left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right)\right| \leq \left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \hat{\boldsymbol{\epsilon}}_j\right|,
\end{aligned} \tag{40}$$

[7] For simplicity, we use the vanilla gradient method. The conclusion can be generalized to the other saliency methods considered in this paper

As is discussed in Section A7, $|(\mathbf{H}_{\mathbf{x}})_{ij}| \leq C_C$. With this, we can have

$$\begin{aligned}
&\left|\sum_{j=1}^{d}(\mathbf{H}_{\mathbf{x}})_{ij}\left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right)\right| \\
\leq &\sum_{j=1}^{d}|(\mathbf{H}_{\mathbf{x}})_{ij}|\left|(\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right| \\
\leq &\sum_{j=1}^{d}C_c\left|(\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right|
\end{aligned} \tag{41}$$

By plugging Eqn. 41 into Eqn. 40, we conclude that as long as $\left|(\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right| \leq \frac{1}{C_c d}\left|\sum_{j=1}^{d}(\mathbf{H}_{\mathbf{x}})_{ij}\hat{\boldsymbol{\epsilon}}_j\right|$, our method could guarantee to improve the robustness against the adversarial perturbations. Let $\delta_i = |E(\hat{\mathbf{x}}, F)_i - E(\mathbf{x}, F)_i|$. If we can ensure that $|\mathbf{x}_i - \tilde{\mathbf{x}}_i| \leq \frac{1}{4C_c d}\delta_i$ and $\left|\hat{\mathbf{x}}_i - \tilde{\hat{\mathbf{x}}}_i\right| \leq \frac{1}{4C_c d}\delta_i$, we can have $\left|\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-\right| \leq \frac{1}{2C_c d}\delta_i$ and $\left|\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-\right| \leq \frac{1}{2C_c d}\delta_i$. Thus, the aforementioned condition can be satisfied, *i.e.,* $\left|(\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right| \leq \frac{1}{C_c d}\delta_i$. By setting $C_2 = \frac{1}{4C_c d}$, we could obtain the robustness conditions in Proposition 1.

□

A9 COROLLARY 1

Consider a multilayer perceptron with L fully-connected hidden layers and a decoy swappable size $K \times 1$. The input of this MLP is $\mathbf{x} \in \mathbb{R}^d$. For each hidden layer, we use the ReLU activation function. Similar to the CNN mentioned above, the output of this CNN is $\mathbf{p} \in \mathbb{R}^C$. The network can be represented as:

$$\begin{aligned}
&\mathbf{m}_\ell = \text{relu}(\mathbf{W}_\ell^T\mathbf{m}_{\ell-1} + \mathbf{b}_\ell), \ \text{ For } \ \ell = 1, 3, ..., L\,, \\
&\mathbf{o} = \mathbf{W}_{L+1}^T\mathbf{m}_L + \mathbf{b}_{L+1}, \\
&\mathbf{p} = \text{softmax}(\mathbf{o})\,.
\end{aligned} \tag{42}$$

where $\mathbf{W}_\ell \in \mathbb{R}^{d_{\ell-1}\times d_\ell}$, for $\ell \in \{1, \cdots, L+1\}$ represents the weights of the neural network, and $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$ represents the biases, where $d_0 = d$ and $d_{L+1} = C$. $\mathbf{m}_\ell \in \mathbb{R}^{d_\ell}$ is the output of each hidden layer, with $\mathbf{m}_0 = \mathbf{x}$ and $\mathbf{o} \in \mathbb{R}^C$ is the logits. The entry-wise softmax operator for target class c is defined as $\mathbf{p}_c = \frac{e^{\mathbf{o}_c}}{\sum_{c'=1}^{C} e^{\mathbf{o}_{c'}}}$, for $c \in \{1, 2, \cdots, C\}$.

Corollary 1. *For the above MLP, Z_i is also bounded by:*

$$Z_i \leq \left|\frac{1}{2}\sum_{k\in\mathcal{K}}(\tilde{\mathbf{x}}_{i+k} - \mathbf{x}_{i+k})(\mathbf{H}_{\mathbf{x}})_{i+k,i}\right| + C_2\,. \tag{43}$$

Proof. Based on the proof of Theorem 1, the gradient of $\mathbf{p}_c$ with respect to $\mathbf{x}$ can be written as follows

$$\nabla_{\mathbf{x}}\mathbf{p}_c = \prod_{l=1}^{L}\mathbf{B}_\ell\mathbf{W}_{L+1}\hat{\mathbf{P}}_{\cdot c}\,. \tag{44}$$

where $\mathbf{B}_\ell = \frac{\partial \mathbf{m}_\ell}{\partial \mathbf{m}_{\ell-1}}$, $\mathbf{B}_\ell \in \mathbb{R}^{d_{\ell-1}\times d_\ell}$. $\hat{\mathbf{P}}_{\cdot c}$ is also defined as $\hat{P} = \text{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$. In the following, we compute $\mathbf{B}_l$. First, we can compute $(\mathbf{B}_1)_{ij}$, in which

$$(\mathbf{B}_1)_{ij} = \frac{\partial(\mathbf{m}_1)_j}{\partial \mathbf{x}_i} = \frac{\partial(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1)_j}{\partial \mathbf{x}_i}\frac{\partial(\mathbf{m}_1)_j}{\partial(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1)_j} = (W_1)_{ij}(a_1)_j\,, \tag{45}$$

where $(a_1)_j = \mathbf{1}\{(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1)_j \geq 0\}$. Similar, we can also compute $(\mathbf{B}_\ell)_{ij}$, for $\ell = 2, 3, ..., L$

$$(\mathbf{B}_\ell)_{ij} = (W_\ell)_{ij}(a_\ell)_j\,, \tag{46}$$

where $(a_\ell)_j = \mathbf{1}\{(\mathbf{W}_\ell^T \mathbf{x} + \mathbf{b}_\ell)_j \geq 0\}$.

Then, we compute the each element in the Hessian matrix $\mathbf{H}_{ij}$. Specifically, based on Eqn. 24, we have

$$\mathbf{H}_{ij} = \left(\sum_{n_L=1}^{d_L} (\prod_{\ell=1}^{L} \mathbf{B}_\ell)_{jn_L} (\mathbf{W}_{L+1})_{n_L \cdot} \right) \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} , \tag{47}$$

where $\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$ is the same with Eqn. 25.

Now, we compute $(\prod_{\ell=1}^{L} \mathbf{B}_l)_{jn_L}$ as

$$(\prod_{\ell=1}^{L} \mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell (\mathbf{B}_L)_{\cdot n_L} , \tag{48}$$

where $(\mathbf{B}_1)_{j\cdot} = [(\mathbf{W}_1)_{j1}(a_1)_1, (\mathbf{W}_1)_{j2}(a_1)_2, ..., (\mathbf{W}_1)_{jd_1}(a_1)_{d_1}]$ and

$$(\mathbf{B}_1)_{j\cdot} \mathbf{B}_2 = [(a_2)_1 \sum_{n_1=1}^{d_1} (C_2)_{1n_1} (a_1)_{n_1}, ..., (a_2)_{d_2} \sum_{n_1=1}^{d_1} (C_2)_{d_2,n_1} (a_1)_{n_1}] , \tag{49}$$

where $(C_2)_{n_2,n_1} = (\mathbf{W}_2)_{n_1,n_2} (\mathbf{W}_1)_{j,n_1}$. For simplicity, we can rewrite $\sum_{n_1=1}^{d_1} (C_2)_{n_2,n_1} (a_1)_{n_1} = (C_2)_{n_2} \sum_{n_1=1}^{d_1} (a_1)_{n_1}$. Then, we have

$$(\mathbf{B}_1)_{j\cdot} \mathbf{B}_2 = [(C_2)_1 (a_2)_1 \sum_{n_1=1}^{d_1} (a_1)_{n_1}, ..., (C_2)_{d_2} (a_2)_{d_2} \sum_{n_1=1}^{d_1} (a_1)_{n_1}] . \tag{50}$$

As such, we can compute $(\mathbf{B}_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell$ as

$$(\mathbf{B}_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell = [(C_{L-1})_1 (a_{L-1})_1 \sum_{\ell=1}^{L-2} \sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell}, ..., (C_{L-1})_{d_{L-1}} (a_{L-1})_{d_{L-1}} \sum_{\ell=1}^{L-2} \sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell}] . \tag{51}$$

Plugging Eqn. 51 into Eqn. 20, we have

$$(\prod_{\ell=1}^{L} \mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell (\mathbf{B}_L)_{\cdot n_L} = (C_L)_{n_L} (a_L)_{n_L} \sum_{\ell=1}^{L-1} \sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell} . \tag{52}$$

Finally, we can obtain that

$$\begin{aligned} \mathbf{H}_{ij} &= \left(\sum_{n_L=1}^{d_L} (C_L)_{n_L} (a_L)_{n_L} \sum_{\ell=1}^{L-1} \sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell} (\mathbf{W}_{L+1})_{n_L \cdot} \right) \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} \\ &= \left(C_j \sum_{\ell=1}^{L} \sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell} \right) \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} , \end{aligned} \tag{53}$$

where C_j is a linear combination of the elements in $(\mathbf{W}_1)_{j\cdot}$, $\mathbf{W}_2$, ..., $\mathbf{W}_{L+1}$.

Note that the Hessian derived from the MLP has a similar form with the Hessian derived from the CNN in Eqn. 30, i.e., the summation of neurons activated by $\mathbf{x}$ multiplying the gradient. Here, the summation of neurons activated by $\mathbf{x}$ is again bounded by the total number of neurons in the network. The gradient $\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$ is bounded by a Lipschitz constant. Similarly, we also have the following inequality for $(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij}$ and $(\mathbf{H}_{\mathbf{x}})_{ij}$, i.e., $|(\mathbf{H}_{\tilde{\mathbf{x}}})_{ij} - (\mathbf{H}_{\mathbf{x}})_{ij}| \leq C_M$.

Similar to Theorem 1,let $\tilde{\mathbf{x}}^+, \tilde{\mathbf{x}}^- \in \left\{ \tilde{\mathbf{x}}^1, \tilde{\mathbf{x}}^2, \cdots, \tilde{\mathbf{x}}^{2n} \right\}$ denotes the decoy which maximizes and minimize $E(\tilde{\mathbf{x}}; F)_i$, respectively. Based on Eqn. 31 to Eqn. 36, we have

$$\left| Z_i - \frac{1}{2} \left| \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) (\mathbf{H}_{\mathbf{x}})_{k,i} \right| \right| \leq C_2 . \tag{54}$$

$C_2 \geq \frac{1}{2} C_M \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) + 2C$. Slightly different for CNN, MLP sometimes is used to process the input that does not have a strong local dependency. In this case, we can set the swappable path size $K = 1$. Then, Eqn. 54 can reformulated as $\left| Z_i - \frac{1}{2} \left| (\tilde{\mathbf{x}}_i^+ - \tilde{\mathbf{x}}_i^-)(\mathbf{H}_{\mathbf{x}})_{i,i} \right| \right| \leq C_2$. As we can observe from this equation, our proposed saliency score is still able to compensate for the gradient saturation problem.

□

Table A1: The hyper-parameter choices of the proposed method on different target models.

	ℓ	λ	patch_size (K)	stride	τ
ImageNet AlexNet	6	10000	3	1	1
ImageNet VGG16	3	10000	3	1	1
ImageNet ResNet	2	10000	3	1	1
SST CNN	2	10000	1	1	1
IDS MLP	2	10000	1	1	1

A10 DATASETS AND EXPERIMENT SETUP

In this section, we introduce the datasets used in our experiments and the neural network trained on each dataset, followed by our choices of hyper-parameters when explaining each model.

ImageNet. We randomly select a subset of samples from the ImageNet validation set, which can be downloaded from the following link: `http://www.image-net.org/`. We adopt the most widely used preprocessing method for the selected images. Specifically, for each image, we resized it to 227×227, converted it to BGR format, and subtract the mean value of each channel [103.939, 116.779, 123.68] from the image. Rather than training our own networks, we downloaded a pretrained VGG16 model, AlexNet model, and ResNet_v1_50 model from the following link: `https://github.com/tensorflow/models/tree/master/research/slim` and `http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/`. We applied our proposed method to explain the predictions of these networks on the selected samples.

SST. We downloaded the Stanford Sentiment Treebank (SST1) from the following link: `https://github.com/harvardnlp/sent-conv-torch/tree/master/data`. The data is spited into a training set of $76,961$ samples and a testing set of $1,821$ samples. We used a pretrained glove embedding to represent each word in the sentences (sample). The embedding of each word is a vector of 100 dimensions. The pretrained embedding matrix can be downloaded from the following link: `http://nlp.stanford.edu/data/wordvecs/glove.6B.zip`. We trained a two-layer CNN with the embeddings as inputs. The model achieves about 80% accuracy on the testing set. The preprocessed testing data and the pretrained model can be downloaded from the following link: `https://tinyurl.com/y9noqj6l`. We run our explanation method on the pretrained model with the testing samples.

Network intrusion detection (IDS). We use a subset of CSE-CIC-IDS2018 dataset (Sharafaldin et al., 2018; for Cybersecurity, 2018), a network intrusion dataset contains the benign network traffic traces and malicious traces generated by three types of attacks: Denial of Service (DoS)-Hulk, SSH-BruteForce, and Infiltration. The training set contains $88,661$ samples and the testing set has $22,165$ samples. Each sample is represented as a vector of 83 dimensions, where each feature represents the statistics of network traffic flows (*e.g.,* Number of packets, Number of bytes, Length of packets, etc). The features are normalized within [0, 1] by using the `scikit-learn` MinMaxScaler function. We trained a two-layer MLP to classify whether an input is a benign traffic or an attack (intrusion). The model reaches 99% accuracy on the testing set. After training the model, we randomly sampled a subset of $2,000$ testing samples and used our method to derive explanations from the model predictions of samples in this subset. The dataset, model, and the descriptions of each feature can be found in `https://tinyurl.com/y9noqj6l`.

Hyper-parameter choices. The hyper-parameter choices of the proposed method on three datasets are shown in Table A1. In the table, ℓ is the index of the layer within the target model that is selected

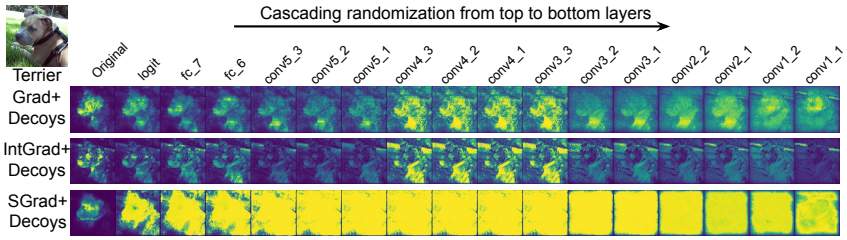


Figure A4: Cascading randomization on VGG16 network. The figure shows the original saliency map (first column) for the terrier. Progression from left to right corresponds to complete randomization of the pretrained VGG16 network weights from the top layer to the bottom layer.

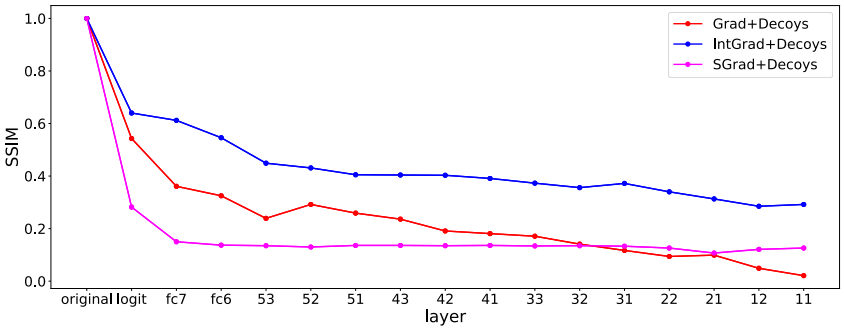


Figure A5: Structural similarity index (SSIM) for Cascading Randomization on VGG16 network.

to generate the decoy images. The hyper-parameter λ controls the weight of $\|F_\ell(\tilde{\mathbf{x}}) - F_\ell(\mathbf{x})\|_\infty$. The patch_size and stride control the size and the stride step of each decoy patch. τ is introduced by Eqn. 8 in Section A6. Note that we set the swappable patch size of SST and IDS data as 1, because their features may not have a strong local correlation. It should also be noted that we selected the swappable patch size of ImageNet data as the widely used convolutional kernel size 3 and stride size 1.

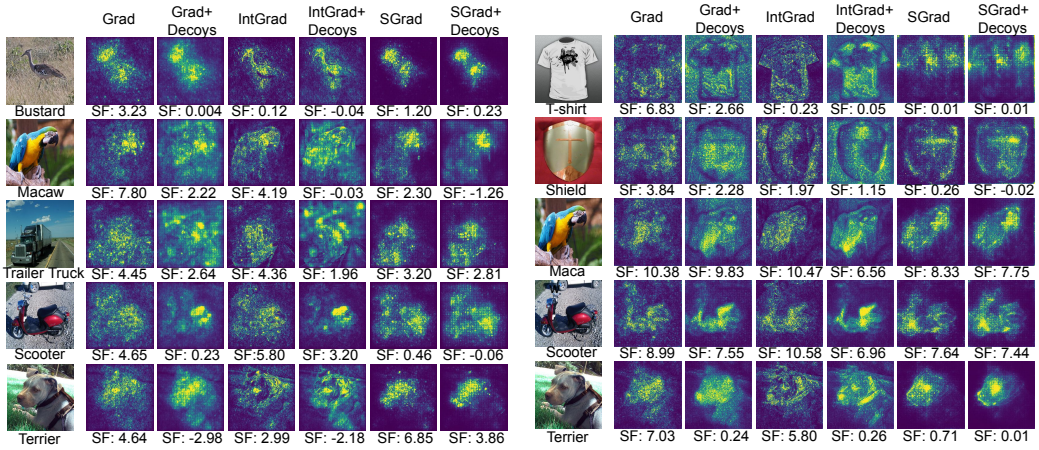
A11 Sanity Check for Decoy-Enhanced Saliency Maps

As suggested by Adebayo et al. (2018), any valid saliency methods should pass the sanity check in the sense that the saliency method should be dependent on the learned parameters of the predictive model, instead of edge or other generic feature detectors. We performed the model parameter randomization test (Adebayo et al., 2018) on the ImageNet dataset by comparing the output of the proposed saliency method on a pretrained VGG16 network with the output of the proposed saliency method on a weight-randomized VGG16 network. If the proposed saliency method indeed depends on the learned parameters of the model, it is expected that the outputs between the two cases differ substantially.

Following the cascading randomization strategy (Adebayo et al., 2018), the weights of pretrained VGG16 network are randomized from the top to bottom layers in a cascading fashion. This cascading randomization procedure is designed to destroy the learned weights successively. As illustrated in Figure A4, the cascading randomization destroys the decoy-enhanced saliency maps combined with three existing saliency methods, qualitatively. The conclusion is also supported by quantitative comparison measured by the structural similarity index (SSIM), shown in Figure A5.

A12 Applicability to other CNN architectures

In addition to the VGG16 model, we generated saliency maps for AlexNet (Krizhevsky et al., 2012) and ResNet (He et al., 2016) trained from the ImageNet dataset. We visualize their saliency maps in Figure A6. We observe that our method consistently outperforms the baseline methods, both



(a) Saliency maps generated on AlexNet. (b) Saliency maps generated on ResNet.

Figure A6: Visualization of saliency maps under different CNN architectures.

Table A2: Quantitative comparison of our method and baselines on the network intrusion dataset. We report the means and standard errors of the fidelity scores.

Dataset	Fidelity (SF)								
	Grad	Grad+Decoys	Grad+Const	IntGrad	IntGrad+Decoys	IntGrad+Const	SGrad	SGrad+Decoys	SGrad+Const
IDS	2.40 ± 3.97	-1.22 ± 6.60	7.97 ± 9.71	13.46 ± 16.01	11.16 ± 14.67	13.23 ± 15.67	8.72 ± 13.05	8.73 ± 13.06	8.83 ± 13.07

quantitatively and qualitatively. Together with the results in Section 4.2.1, these results suggest that we can apply our decoy-enhanced saliency methods to various feed-forward network architectures and expect consistent performance.

A13 Performances on the Network Intrusion Dataset.

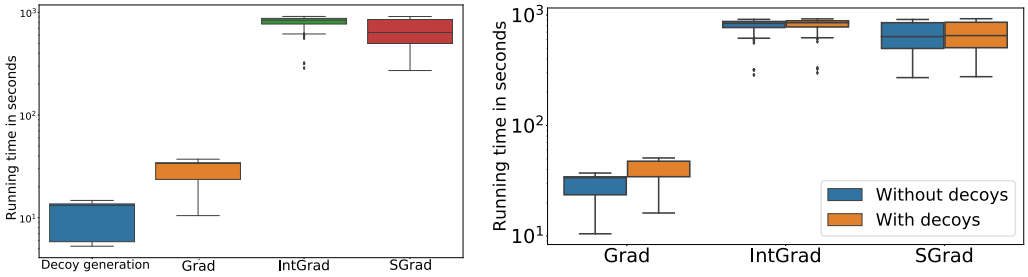
Rather than visualizing the saliency scores through heatmaps, we apply the following to compare the saliency scores obtained by different methods qualitatively. We ranked the features based on their saliency scores and compared the ranking obtained by the existing methods with that obtained by our decoy-enhanced method. "Minimum size of packet in forward direction", "Minimum length of a packet", "Minimum time between two packets sent in the forward direction" are ranked higher by our methods than the baselines. These features could capture the differences between benign and malicious traffics. This is because attackers usually tend to rapidly send small packages to discover the backdoors in the victim network system, while the benign users may send much larger packages with a longer interval between two packages. On the contrary, features that are not that useful for intrusion detection (*e.g.,* timestamp, Download and upload ratio) are wrongly pinpointed by the existing method. However, our methods correctly assign lower importance to these features. Table A2 shows the fidelity comparisons of different saliency methods. We can observe that our decoys-enhanced methods outperform the original saliency methods. These results show that our method could pinpoint more accurate features and achieve a higher fidelity than baselines. We also evaluated the constant-decoy baseline, i.e., generating decoys by replacing each feature with its mean value. The results in Table A2 shows that constant-decoy cannot achieve comparable performance with our method, and even perform worse than the existing methods in some cases. In summary, the results on this dataset are aligned with those on the other datasets, confirming our method's applicability to multilayer perceptrons.

A14 Decoys on the Variants of Integrated Gradient.

In Section 4, we applied the original integrated gradient. Recent research (Sturmfels et al., 2020) suggests some variants that improve the performance of the integrated gradient. Here, by using ImageNet data, we evaluate whether our decoy method could further improve the variants of the

Table A3: Quantitative comparison of our method and baselines on the network intrusion dataset. We report the means and standard errors of the fidelity scores.

Saliency methods	Fidelity (SF)	
	Without decoys	With decoys
Expected Gradient	2.49 ± 1.81	$\mathbf{0.35 \pm 1.33}$
Integrated Gradient (baseline: uniform)	3.38 ± 2.46	$\mathbf{0.50 \pm 1.58}$
Integrated Gradient (baseline: zero)	3.69 ± 2.64	$\mathbf{0.93 \pm 1.19}$



(a) Run time to optimize *one* decoy and calculate saliency map with three existing methods.

(b) Run time to compute saliency maps with and without optimizing *one* decoy.

Figure A7: Run time of decoy generation. The comparison is conducted in the same CPU/GPU to ensure fairness. Note that "Grad", "IntGrad", and "SGrad" stands for the vanilla gradient, the integrated gradient, and the SmoothGrad, respectively.

integrated gradient. Specifically, we consider two variants mentioned in Sturmfels et al. (2020): integrated gradient with uniform baseline and Expected Gradient.

Table A3 shows the comparison of explanation fidelity of each method with/without decoys. As is depicted in the table, our decoy method consistently improves the fidelity of the different variants of the integrated gradient (Column 2 vs. Column 3). Together with the results on the vanilla gradient and SmoothGrad, they demonstrate the generalizability of our technique to different saliency methods. As part of future work, we will investigate the effectiveness of adding our technique to more variants of our selected base saliency methods mentioned in Sturmfels et al. (2020).

A15 Runtime of Decoy Generations

To evaluate the computational cost of our decoy generations, we carried out the run time comparison between optimizing one decoy and calculating three types of saliency methods, repeated 500 times with respect to different patch masks. As illustrated in Figure A7a, on average, optimizing one decoy is 62.3% faster than the fastest vanilla gradient-based saliency method. For other methods, the optimization is even less expensive, in a relative sense.

As is mentioned in Section 3.4, our decoy generation, and the saliency map computation can be run parallelly in a batch mode. In the optimal case, where we have enough resources to compute each decoy saliency map (*i.e.,* $E(\tilde{\mathbf{x}}^i; F)$) parallelly, the overall runtime of generating one decoy-enhanced saliency map is the total time of generating one decoy and computing one saliency map using the existing methods. Figure A7b shows the comparisons between the optimal decoy-enhanced saliency map generation time and the original saliency map generation time across three saliency methods. As we can observe from the figure, our method introduces negligible computational overhead over the existing methods. Taking a step back, when the users have limited resources for running the decoy-enhanced saliencies in a fully parallel fashion. Since the decoy generation is faster than the existing methods, in most cases, partial parallel computing still does not significantly increase the computational overhead. In our experiment, we used a server with 32 CPU cores and 8 Nvidia 1080 Ti GPUs. Our method introduces about 2X$\sim$5X overheard to the existing saliency methods. We argue that for an ensemble method, this overhead is acceptable. In addition, our method can be exponentially faster on more powerful machines, which escalates the practicality of our method.

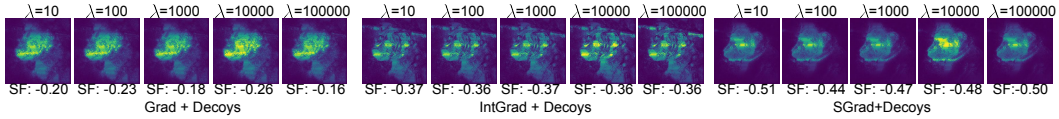


Figure A8: Visualization of saliency maps optimized using different initial λ.

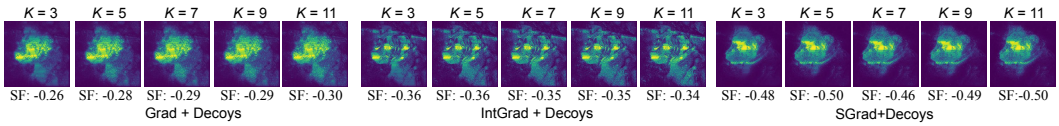


Figure A9: Visualization of saliency maps optimized using different patch size K.

A16 HYPER-PARAMETER SENSITIVITY

We also conduct experiments on the VGG16 to understand the impact of hyper-parameter choices on the performance of our optimization-based decoy generation method. Specifically, we focus on the choice of three hyper-parameters: network layer ℓ, initial Lagrange multiplier λ, and patch size.

Accordingly, we first varied the value of ℓ for VGG16 and compared the differences of the generated decoy saliencies from the three aforementioned saliency methods. In particular, we set it to range from the first convolutional layer to the last pooling layer and demonstrate the generated decoy saliencies in Figure A13. Note that according to our design, only the convolutional layers and the pooling layers can be used to generate decoy images. For each saliency method, Figure A13 demonstrates that the decoy saliencies generated from different layers for the same image are of similar qualities. Figure A13 also shows the mean and standard derivation of the SF scores for each saliency method. These quantitative results also support the conclusion that our approach is not sensitive to the layer. This is likely because, as previous research has shown (Chan et al., 2015; Saxe et al., 2011), the final classification results of a DNN are not highly related to the hidden representations. As a result, generating decoy saliencies for the same sample with the same label from different layers should yield similar results.

We also varied the initial Lagrange multiplier λ to be $\left\{10^1, 10^2, 10^3, 10^4, 10^5\right\}$ and compared the differences of the generated decoy saliencies. Figure A8 depicts the quantitative and qualitative comparison results. As shown in the figure, the different choices of initial λ all produce similar saliency maps, indicating a negligible influence upon our method.

Finally, we increased the patch size to be $\{3, 5, 7, 9, 11\}$ and showed the generated decoy saliencies in Figure A9. The results show that varying the patch size within a certain range only imposes a negligible influence upon our method. *Note that, as is mentioned in Section 3.4, we generate decoy masks by sliding the swappable patch across a given input. This means that given a constant stride* 1*, the number of decoys* n *is unique to the patch size* K*. Specifically,* n *is inversely proportional to* K*. As such, we do not explicitly set* n *as our hyper-parameter and control it by adjusting* K*. Given that our method is not sensitive to the choice of* K*, it also not sensitive to the variation of* n*.*

The results in Figure A13, A8, and A9 indicate we can expect to obtain stable decoy saliencies when the hyper-parameters are subtly varied. This is a critical characteristic because users do not need to overly worry about setting very precise hyper-parameters to obtain a desired saliency map.

A17 OBJECT LOCALIZATION

Table A4: ImageNet localization accuracy on VGG16 network using different thresholding strategies.

Accuracy	Value thresholding (0.25)	Energy thresholding (0.25)	Mean thresholding (0.25)
Gradient	0.662	0.715	0.662
Gradient+Decoys	**0.722**	**0.723**	**0.665**

We compare our method and the vanilla gradient method on the object localization task (Dabkowski & Gal, 2017; Fong & Vedaldi, 2017), where the model was trained with the class label only without access to any localization data. We carried out Imagenet ILSVRC'14 localization task (Russakovsky et al., 2015) which contains 50K ImageNet validation images with annotated bounding boxes as ground truth. For each of the image in the dataset, we first calculated the gradient-based saliency maps with and without using blurry decoys, based on pretrained VGG16 network. Following the preprocessing steps suggested by Dabkowski & Gal (2017); Fong & Vedaldi (2017), for each calculated saliency maps, a bounding box for the most dominant object is predicted by thresholding the corresponding saliency map and extracting the tightest bounding box that contains the whole remaining saliency map.

We investigated three different thresholding strategies suggested by Fong & Vedaldi (2017), including value thresholding, energy thresholding, and mean thresholding. Following the evaluation protocol suggested by Dabkowski & Gal (2017); Fong & Vedaldi (2017), the extracted localization box has to have Intersect over Union (IoU) greater than 0.5 with any of the ground truth bounding boxes in order to consider the localization successful, otherwise it is regarded as failure. As shown in Table A4, in terms of accuracy, decoy-enhanced saliency maps perform better than vanilla saliency maps without decoys.

A18 Additional experimental results

Figure A11, Figure A10, and Figure A12 provide more results of the fidelity and robustness evaluation. These results are consistent with those shown in the Section 4.

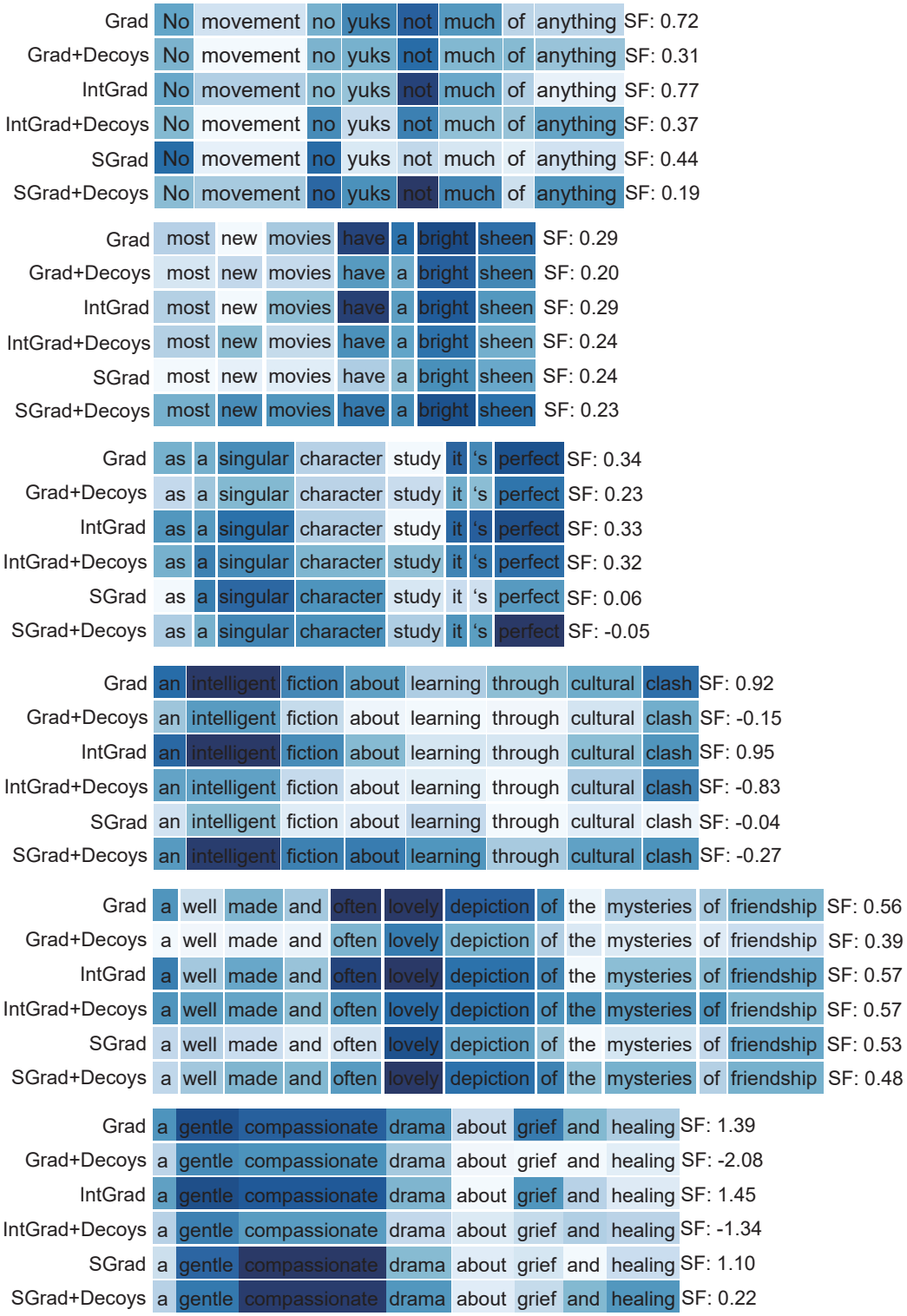


Figure A10: Visualization of saliency maps on the sentences in SST dataset.

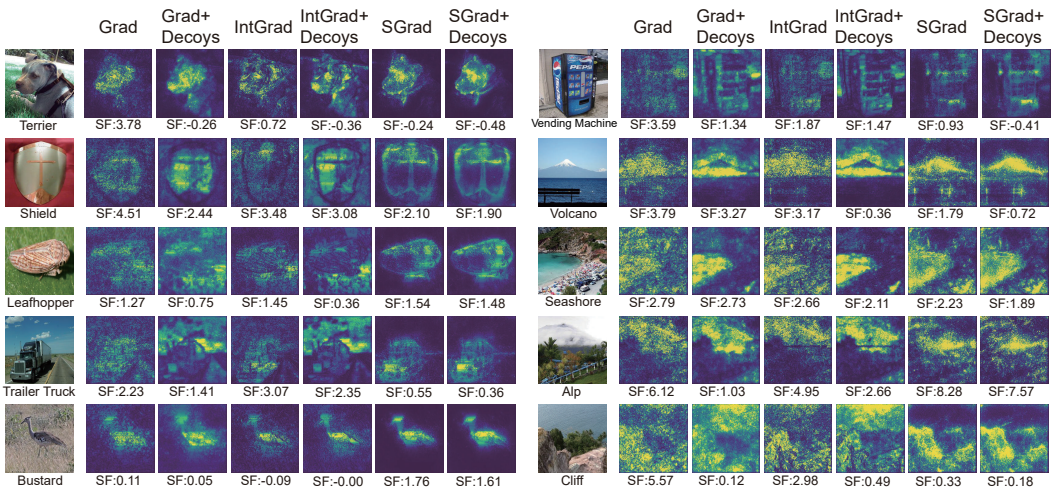


Figure A11: Visualization of saliency maps on the images in ImageNet dataset.

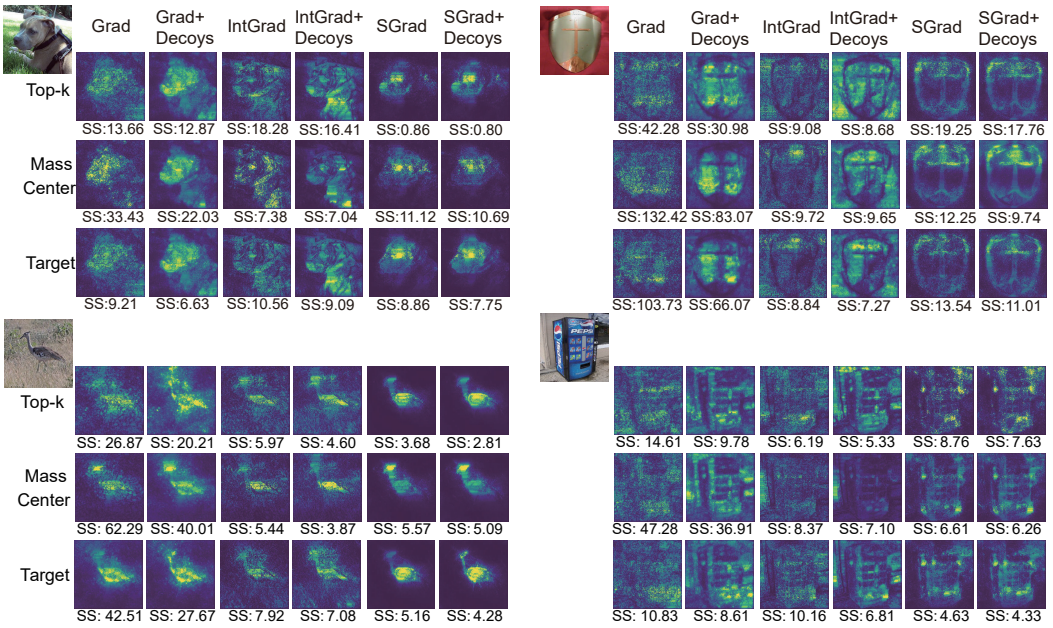
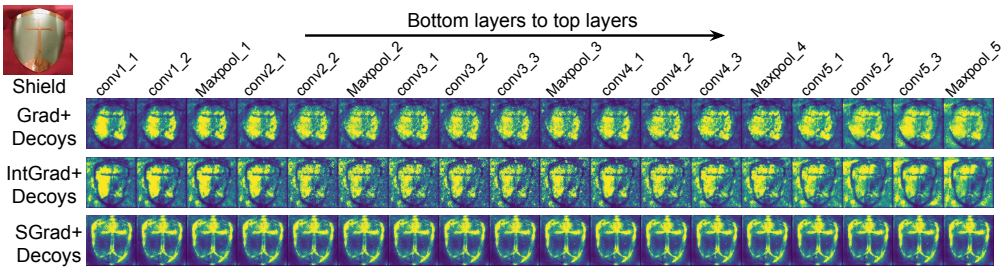
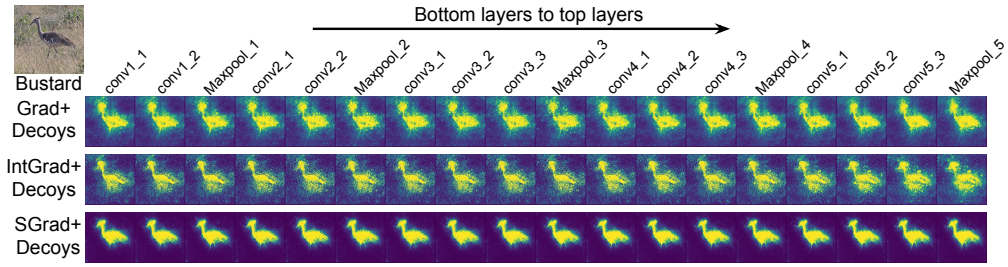


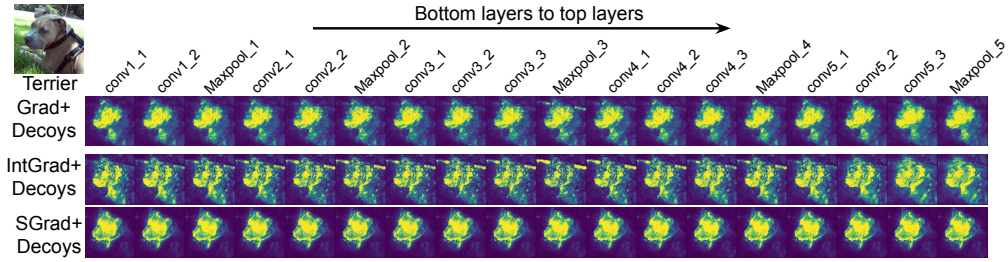
Figure A12: Visualization of saliency maps on the perturbed images generated by using three attacks in VGG16.



(a) The mean and standard derivation of SF score for gradient, integrated gradient and SmoothGrad are: (-0.01, 0.01), (-0.03, 0.01), (0.09, 0.01).



(b) The mean and standard derivation of SF score for gradient, integrated gradient and SmoothGrad are: (3.34, 0.73), (2.21, 1.33), (1.49, 0.18).



(c) The mean and standard derivation of SF score for gradient, integrated gradient and SmoothGrad are: (-0.11, 0.02), (-0.18, 0.02), (-0.21, 0.01).

Figure A13: Demonstrations of decoy-enhanced saliency maps generated from each convolutional and pooling layer in VGG16.