DANCE: Enhancing saliency maps using decoys

S1 Implementation details

S1.1 Aggregating multiple decoy patches into single decoy mask

For example, given an input image $\mathbf{x} \in \mathcal{R}^{\sqrt{d}\times\sqrt{d}}$ and a swappable patch with size P, we obtain $(\sqrt{d} - P + \text{stride})^2$ unique masks by sliding the swappable patch across the input. For an input with high dimensionality and relatively small patch size, this n will be relatively large. In our implementation, to reduce the computational cost, we aggregate m ($m < n$) masks into one combined mask, which contains m swappable patches at different locations. Then, we generate a decoy by taking this combined mask as the input. The optimization process will perturb all the features within these m swappable patches. With this aggregation, the decoy mask number reduces to $n = \left\lfloor (\sqrt{d} - P + \text{stride})^2/m \right\rfloor$.

S1.2 Optimization 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 the Eqn.(2) of Section 3.3. 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.,* $\tilde{\mathbf{x}} \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]^d$). 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:

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

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:

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

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 $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$. Note that Eqn. (2) 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. Note that, when solving decoys, before applying the gradient descent, we add a small perturbation to the input via random initialization by following the insight of SmoothGrad. This helps avoid the zero gradients of saturated inputs and obtain meaningful decoy perturbations.

S2 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*

$$\left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| \leq C \,. \tag{3}$$

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

$$F^c(\mathbf{x}) \approx F^c(\tilde{\mathbf{x}}) + \nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}})^T \Delta + \frac{1}{2} \Delta^T \mathbf{H}_{\tilde{\mathbf{x}}} \Delta \,, \tag{4}$$

where $\Delta = \mathbf{x} - \tilde{\mathbf{x}}$. By definition of the decoys in Section 3.2 (*i.e.,* $F^c(\mathbf{x}) = F^c(\tilde{\mathbf{x}})$), the following equation holds:

$$\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}})^T \Delta \approx -\frac{1}{2} \Delta^T \mathbf{H}_{\tilde{\mathbf{x}}} \Delta \,. \tag{5}$$

From the above equation, we can see that, for a linear model, the linearity zeroes out the gradient of the decoys, causing our method to output zero saliency scores for all input features. We clarified in Section 5 that our method is mainly defined for non-linear complicated models.

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

$$\begin{aligned} \nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}})^T \Delta &= \sum_{i \in \mathcal{K}} (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i (\mathbf{x}_i - \tilde{\mathbf{x}}_i) \,, \\ \Delta^T \mathbf{H}_{\tilde{\mathbf{x}}} \Delta &= \sum_{i \in \mathcal{K}} (\mathbf{x}_i - \tilde{\mathbf{x}}_i) \sum_{k \in \mathcal{K}} (\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} (\mathbf{x}_k - \tilde{\mathbf{x}}_k) \,. \end{aligned} \tag{6}$$

Plugging Eqn. (6) into Eqn. (5), we have

$$\sum_{i \in \mathcal{K}} [(\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i + \frac{1}{2} \sum_{k \in \mathcal{K}} (\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} (\mathbf{x}_k - \tilde{\mathbf{x}}_k)](\mathbf{x}_i - \tilde{\mathbf{x}}_i) = 0 \,. \tag{7}$$

Then we can derive

$$\begin{aligned} \left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i + \frac{1}{2} \sum_{k \in \mathcal{K}} (\mathbf{x}_k - \tilde{\mathbf{x}}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| &\leq C \,, \\ \left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| &\leq C \,. \end{aligned} \tag{8}$$

First, we can derive $|\tilde{\mathbf{x}}_i - \mathbf{x}_i|$ is bounded by $2\text{max}(\mathbf{x}_{\text{max}}, |\mathbf{x}_{\text{min}}|)$. 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). [1] 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

[1] 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 $\ell = 1, 2, ..., L$.

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

$$\begin{aligned} &\mathbf{m}_\ell = \text{pool}(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})) \quad \text{for} \quad \ell = 1, 2, 3, ..., L\,, \\ &\mathbf{o} = \mathbf{W}_{L+1}^T \mathbf{m}_L + \mathbf{b}_{L+1}, \\ &\mathbf{p} = \text{softmax}(\mathbf{o})\,, \end{aligned} \tag{9}$$

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*

$$\left| Z_i - \frac{1}{2} \left| \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_\mathbf{x})_{k,i} \right| \right| \leq C_1\,. \tag{10}$$

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:

$$\nabla_\mathbf{x} \mathbf{p}_c = \prod_{\ell=1}^{L} \frac{\partial \mathbf{m}_\ell}{\partial \mathbf{m}_{\ell-1}} \frac{\partial \mathbf{o}}{\partial \mathbf{m}_L} \frac{\partial \mathbf{p}_c}{\partial \mathbf{o}}, \tag{11}$$

where

$$\frac{\partial \mathbf{o}}{\partial \mathbf{m}_L} = \mathbf{W}_{L+1}\,, \tag{12}$$

and

$$\begin{cases} \frac{\partial \mathbf{p}_c}{\partial \mathbf{o}_{c'}} = (\mathbf{p}_c - \mathbf{p}_c^2) & \text{if } c' = c\,, \\ \frac{\partial \mathbf{p}_c}{\partial \mathbf{o}_{c'}} = -\mathbf{p}_c \mathbf{p}_{c'} & \text{otherwise}\,. \end{cases} \tag{13}$$

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

$$\frac{\partial \mathbf{p}_c}{\partial \mathbf{o}} = \hat{\mathbf{P}}_{\cdot c}\,, \tag{14}$$

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

$$\begin{cases} \frac{\partial (\mathbf{m}_\ell)_j}{\partial (\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n} = 1 & \text{if } \hat{j} - n \in \mathcal{J}_\ell, \text{ and } n = \text{argmax}_{n' \in \hat{j} + \mathcal{J}_\ell} (\mathbf{g}_\ell * \mathbf{m}_{\ell-1})_{n'}\,, \\ \frac{\partial (\mathbf{m}_\ell)_j}{\partial (\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n} = 0 & \text{otherwise}\,, \end{cases} \tag{15}$$

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

$$\begin{cases} \frac{\partial (\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n}{\partial (\mathbf{m}_{\ell-1})_i} = (a_\ell)_n (\mathbf{g}_\ell)_{n-i} & \text{if } n - i \in \mathcal{J}_\ell\,, \\ \frac{\partial (\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1}))_n}{\partial (\mathbf{m}_{\ell-1})_i} = 0 & \text{otherwise}\,, \end{cases} \tag{16}$$

where $(a_\ell)_n = \mathbf{1}\{(\text{relu}(\mathbf{g}_\ell * \mathbf{m}_{\ell-1})_n) \geq 0\}$. If we change the activation function to either sigmoid or tanh, then $(a_\ell)_n$ in Eqn. (16) 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. (15) with (16), we have

$$\begin{cases} (\mathbf{B}_\ell)_{ij} = \frac{\partial (\mathbf{m}_\ell)_j}{\partial (\mathbf{m}_{\ell-1})_i} = (a_\ell)_n (\mathbf{g}_\ell)_{n-i} & \text{if } n - i \in \mathcal{J}_\ell,\ \hat{j} - n \in \mathcal{J}_\ell, \text{ and } n = \text{argmax}_{n' \in \hat{j} + \mathcal{J}_\ell} (\mathbf{g}_\ell * \mathbf{m}_{\ell-1})_{n'}\,, \\ (\mathbf{B}_\ell)_{ij} = \frac{\partial (\mathbf{m}_\ell)_j}{\partial (\mathbf{m}_{\ell-1})_i} = 0 & \text{otherwise}\,. \end{cases} \tag{17}$$

For simplicity, we rewrite the non-zero condition as $n \in \hat{\mathcal{J}}_\ell$. Plugging $\mathbf{B}_\ell$, $\ell = 1, ..., L$, into Eqn. 11, 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:

$$\begin{aligned}\mathbf{H}_{ij} = \nabla_{\mathbf{x}_i}(\nabla_{\mathbf{x}_j}\mathbf{p}_c) &= \frac{\partial(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{j\cdot}\mathbf{W}_{L+1}\hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} \\ &= (\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{j\cdot}\mathbf{W}_{L+1}\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} = \left(\sum_{n_L=1}^{d_L}(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L}(\mathbf{W}_{L+1})_{n_L\cdot}\right)\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}\,,\end{aligned} \tag{18}$$

and

$$\frac{\partial \hat{\mathbf{P}}_{c'c}}{\partial \mathbf{x}_i} = \begin{cases} (1-2\mathbf{p}_c)\nabla_{\mathbf{x}_i}\mathbf{p}_c & \text{if } c' = c\,, \\ \mathbf{p}_c\nabla_{\mathbf{x}_i}\mathbf{p}_{c'} + \mathbf{p}_{c'}\nabla_{\mathbf{x}_i}\mathbf{p}_c & \text{otherwise}\,. \end{cases} \tag{19}$$

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

$$(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell(\mathbf{B}_L)_{\cdot n_L}\,, \tag{20}$$

where

$$(\mathbf{B}_1)_{j\cdot}\mathbf{B}_2 = [0, ..., C_{n_2}(a_2)_{n_2}\sum_{n_1\in\hat{\mathcal{J}}_1}(a_1)_n\mathbf{g}_{n-1}, ..., 0]\,, \tag{21}$$

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

$$(\mathbf{B}_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell = [0, .., C_{n_{L-1}}(a_{L-1})_{n_{L-1}}\sum_{\ell=1}^{L-2}\sum_{n_\ell\in\hat{\mathcal{J}}_\ell}(a_\ell)_{n_\ell}, ..., 0]\,. \tag{22}$$

Plugging Eqn. (22) into Eqn. (20), we have

$$(\prod_{\ell=1}^{L}\mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot}\prod_{\ell=2}^{L-1}\mathbf{B}_\ell(\mathbf{B}_L)_{\cdot n_L} = (C_L)_{n_L}(a_L)_{n_L}\sum_{\ell=1}^{L-1}\sum_{n_\ell\in\hat{\mathcal{J}}_\ell}(a_\ell)_{n_\ell}\,. \tag{23}$$

Plugging Eqn. (23) into Eqn. (18), we have

$$\mathbf{H}_{ij} = \left(C_j\sum_{\ell=1}^{L}\sum_{n_\ell\in\hat{\mathcal{J}}_\ell}(a_\ell)_{n_\ell}\right)\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}\,, \tag{24}$$

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 $\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$. *The first part shows that the Hessian includes the neighborhood features that are jointly activated, indicating inter-feature interaction.*

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. (19). 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.[2]

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

$$\left| (\nabla_{\tilde{\mathbf{x}}} F^c(\tilde{\mathbf{x}}))_i - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}})_{i,k} \right| \leq C \,, \tag{25}$$

Then, we can derive

$$\frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} - C \leq (\nabla_{\tilde{\mathbf{x}}^+} F^c(\tilde{\mathbf{x}}^+))_i \leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} + C \,, \tag{26}$$

$$-\frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} - C \leq -(\nabla_{\tilde{\mathbf{x}}^-} F^c(\tilde{\mathbf{x}}^-))_i \leq -\frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} + C \,, \tag{27}$$

Then, we have

$$\begin{aligned}
Z_i &= (\nabla_{\tilde{\mathbf{x}}^+} F^c(\tilde{\mathbf{x}}^+))_i - (\nabla_{\tilde{\mathbf{x}}^-} F^c(\tilde{\mathbf{x}}^-))_i \\
&\leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} + 2C \\
&\leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)((\mathbf{H}_{\mathbf{x}})_{i,k} + C_C) - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)((\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} - C_C) + 2C \\
&\leq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{i,k} + \frac{1}{2} C_C \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) + 2C \,,
\end{aligned} \tag{28}$$

And

$$\begin{aligned}
Z_i &= (\nabla_{\tilde{\mathbf{x}}^+} F^c(\tilde{\mathbf{x}}^+))_i - (\nabla_{\tilde{\mathbf{x}}^-} F^c(\tilde{\mathbf{x}}^-))_i \\
&\geq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^+})_{i,k} - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)(\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} - 2C \\
&\geq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \mathbf{x}_k)((\mathbf{H}_{\mathbf{x}})_{i,k} - C_C) - \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^- - \mathbf{x}_k)((\mathbf{H}_{\tilde{\mathbf{x}}^-})_{i,k} + C_C) + 2C \\
&\geq \frac{1}{2} \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{i,k} - \frac{1}{2} C_C \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) - 2C \,,
\end{aligned} \tag{29}$$

Combining Eqn. (28) with Eqn. (29), we have

$$\left| Z_i - \frac{1}{2} \left| \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{k,i} \right| \right| \leq C_1 \,. \tag{30}$$

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 $C_1 \geq \frac{1}{2} C_C \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) + 2C$. Note that this upper bound is data specific, and we leave the exploration on its tightness as a part of future works.

□

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

S3 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 and* $\delta_i = |E(\hat{\mathbf{x}}, F)_i - E(\mathbf{x}, F)_i|$*, then the following relation can be guaranteed.*

$$|(Z_{\hat{\mathbf{x}}})_i - (Z_{\mathbf{x}})_i| \leq |(E(\hat{\mathbf{x}}, F)_i - E(\mathbf{x}, F))_i| \,. \tag{31}$$

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. [3] Given both saliency maps, we can further compute the difference between $E(\hat{\mathbf{x}}, F)$ and $E(\mathbf{x}, F)$ as

$$(E(\hat{\mathbf{x}}, F) - E(\mathbf{x}, F))_i = \nabla_{\hat{\mathbf{x}}} F^c(\hat{\mathbf{x}}) - \nabla_{\mathbf{x}} F^c(\mathbf{x}) = (\mathbf{H}_{\mathbf{x}}(\hat{\mathbf{x}} - \mathbf{x}))_i = \sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \hat{\boldsymbol{\epsilon}}_j \,. \tag{32}$$

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 S2, 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 i^{th} 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

$$\begin{aligned}
&(Z_{\hat{\mathbf{x}}})_i - (Z_{\mathbf{x}})_i \\
=&\left(E(\tilde{\hat{\mathbf{x}}}^+, F)_i - E(\tilde{\hat{\mathbf{x}}}^-, F)_i\right) - \left(E(\tilde{\mathbf{x}}^+, F)_i - E(\tilde{\mathbf{x}}^-, F)_i)\right) \\
=&\left((\mathbf{H}_{\mathbf{x}}(\tilde{\hat{\mathbf{x}}}^+ - \mathbf{x}))_i - (\mathbf{H}_{\mathbf{x}}(\tilde{\hat{\mathbf{x}}}^- - \mathbf{x}))_i\right) - \left((\mathbf{H}_{\mathbf{x}}(\tilde{\mathbf{x}}^+ - \mathbf{x}))_i - (\mathbf{H}_{\mathbf{x}}(\tilde{\mathbf{x}}^- - \mathbf{x}))_i\right) \\
=&\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right) \,.
\end{aligned} \tag{33}$$

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,

$$\begin{aligned}
&\left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right)\right| - \left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \hat{\boldsymbol{\epsilon}}_j\right| \leq 0, \\
&\left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right)\right| \leq \left|\sum_{j=1}^{d} (\mathbf{H}_{\mathbf{x}})_{ij} \hat{\boldsymbol{\epsilon}}_j\right|,
\end{aligned} \tag{34}$$

[3] 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 S2, $|(\mathbf{H}_{\mathbf{x}})_{ij}| \leq C_C$. With this, we can have

$$\begin{aligned}
&\left|\sum_{j=1}^{d}(\mathbf{H}_{\mathbf{x}})_{ij}\left((\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right)\right| \\
\leq &\sum_{j=1}^{d}|(\mathbf{H}_{\mathbf{x}})_{ij}|\left|(\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right| \\
\leq &\sum_{j=1}^{d}C_c\left|(\tilde{\hat{\mathbf{x}}}_j^+ - \tilde{\hat{\mathbf{x}}}_j^-) - (\tilde{\mathbf{x}}_j^+ - \tilde{\mathbf{x}}_j^-)\right|
\end{aligned} \tag{35}$$

By plugging Eqn. (35) into Eqn. (34), 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.

□

S4 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:

$$\begin{aligned}
&\mathbf{m}_\ell = \text{relu}(\mathbf{W}_\ell^T\mathbf{m}_{\ell-1} + \mathbf{b}_\ell), \quad \text{For} \quad \ell = 1, 3, ..., L\,, \\
&\mathbf{o} = \mathbf{W}_{L+1}^T\mathbf{m}_L + \mathbf{b}_{L+1}, \\
&\mathbf{p} = \text{softmax}(\mathbf{o})\,.
\end{aligned} \tag{36}$$

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:*

$$Z_i \leq \left|\frac{1}{2}\sum_{k\in\mathcal{K}}(\tilde{\mathbf{x}}_{i+k} - \mathbf{x}_{i+k})(\mathbf{H}_{\mathbf{x}})_{i+k,i}\right| + C_2\,. \tag{37}$$

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

$$\nabla_{\mathbf{x}}\mathbf{p}_c = \prod_{l=1}^{L}\mathbf{B}_\ell\mathbf{W}_{L+1}\hat{\mathbf{P}}_{\cdot c}\,. \tag{38}$$

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

$$(\mathbf{B}_1)_{ij} = \frac{\partial(\mathbf{m}_1)_j}{\partial \mathbf{x}_i} = \frac{\partial(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1)_j}{\partial \mathbf{x}_i}\frac{\partial(\mathbf{m}_1)_j}{\partial(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1)_j} = (W_1)_{ij}(a_1)_j\,, \tag{39}$$

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 $\ell = 2, 3, ..., L$

$$(\mathbf{B}_\ell)_{ij} = (W_\ell)_{ij}(a_\ell)_j \,, \tag{40}$$

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. (18), we have

$$\mathbf{H}_{ij} = \left(\sum_{n_L=1}^{d_L} (\prod_{\ell=1}^{L} \mathbf{B}_\ell)_{jn_L} (\mathbf{W}_{L+1})_{n_L \cdot} \right) \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} \,, \tag{41}$$

where $\frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i}$ is the same with Eqn. (19).

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

$$(\prod_{\ell=1}^{L} \mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell (\mathbf{B}_L)_{\cdot n_L} \,, \tag{42}$$

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

$$(\mathbf{B}_1)_{j\cdot}\mathbf{B}_2 = [(a_2)_1 \sum_{n_1=1}^{d_1} (C_2)_{1n_1}(a_1)_{n_1}, ..., (a_2)_{d_2} \sum_{n_1=1}^{d_1} (C_2)_{d_2,n_1}(a_1)_{n_1}] \,, \tag{43}$$

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

$$(\mathbf{B}_1)_{j\cdot}\mathbf{B}_2 = [(C_2)_1(a_2)_1 \sum_{n_1=1}^{d_1} (a_1)_{n_1}, ..., (C_2)_{d_2}(a_2)_{d_2} \sum_{n_1=1}^{d_1} (a_1)_{n_1}] \,. \tag{44}$$

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

$$(\mathbf{B}_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell = [(C_{L-1})_1(a_{L-1})_1 \sum_{\ell=1}^{L-2}\sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell}, ..., (C_{L-1})_{d_{L-1}}(a_{L-1})_{d_{L-1}} \sum_{\ell=1}^{L-2}\sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell}] \,. \tag{45}$$

Plugging Eqn. (45) into Eqn. (14), we have

$$(\prod_{\ell=1}^{L} \mathbf{B}_\ell)_{jn_L} = (B_1)_{j\cdot} \prod_{\ell=2}^{L-1} \mathbf{B}_\ell (\mathbf{B}_L)_{\cdot n_L} = (C_L)_{n_L}(a_L)_{n_L} \sum_{\ell=1}^{L-1}\sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell} \,. \tag{46}$$

Finally, we can obtain that

$$\begin{aligned} \mathbf{H}_{ij} &= \left(\sum_{n_L=1}^{d_L} (C_L)_{n_L}(a_L)_{n_L} \sum_{\ell=1}^{L-1}\sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell} (\mathbf{W}_{L+1})_{n_L \cdot} \right) \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} \\ &= \left(C_j \sum_{\ell=1}^{L}\sum_{n_\ell=1}^{d_\ell} (a_\ell)_{n_\ell} \right) \frac{\partial \hat{\mathbf{P}}_{\cdot c}}{\partial \mathbf{x}_i} \,, \end{aligned} \tag{47}$$

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. 24, 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. (25) to Eqn. (30), we have

$$\left| Z_i - \frac{1}{2} \left| \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-)(\mathbf{H}_{\mathbf{x}})_{k,i} \right| \right| \leq C_2 \,. \tag{48}$$

$C_2 \geq \frac{1}{2} C_M \sum_{k \in \mathcal{K}} (\tilde{\mathbf{x}}_k^+ - \tilde{\mathbf{x}}_k^-) + 2C$. 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. (48) 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 S1: The hyper-parameter choices of the proposed method on different target models.

	ℓ	λ	patch_size (P)	stride	τ	m
ImageNet AlexNet	6	10000	3	1	1	100
ImageNet VGG16	3	10000	3	1	1	100
ImageNet ResNet	2	10000	3	1	1	100
SST CNN	2	10000	1	1	1	1
IDS MLP	2	10000	1	1	1	1

S5 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

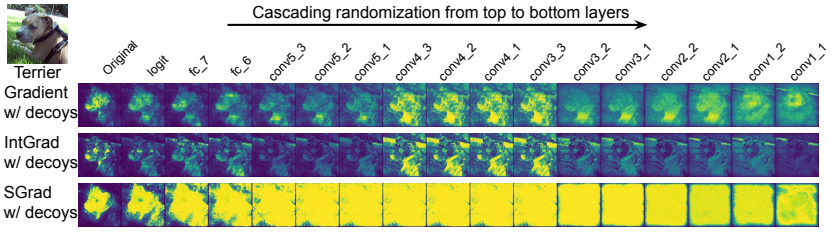


Figure S1: Cascading randomization on VGG16 network. The figure shows the original saliency map (first column) for the terrier. Progression from left to right corresponds to complete randomization of the pretrained VGG16 network weights from the top layer to the bottom layer. Note that, here, we followed the visualization method in Adebayo et al. (2018) to show the saliency maps, i.e., 0-1 normalization.

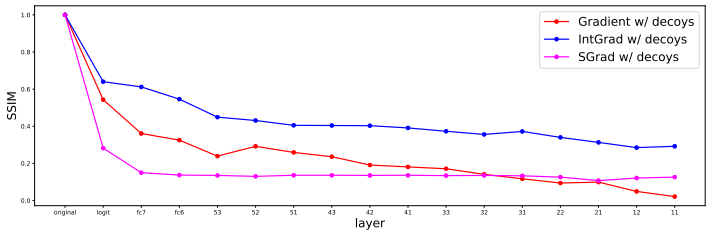


Figure S2: Structural similarity index (SSIM) for Cascading Randomization on VGG16 network.

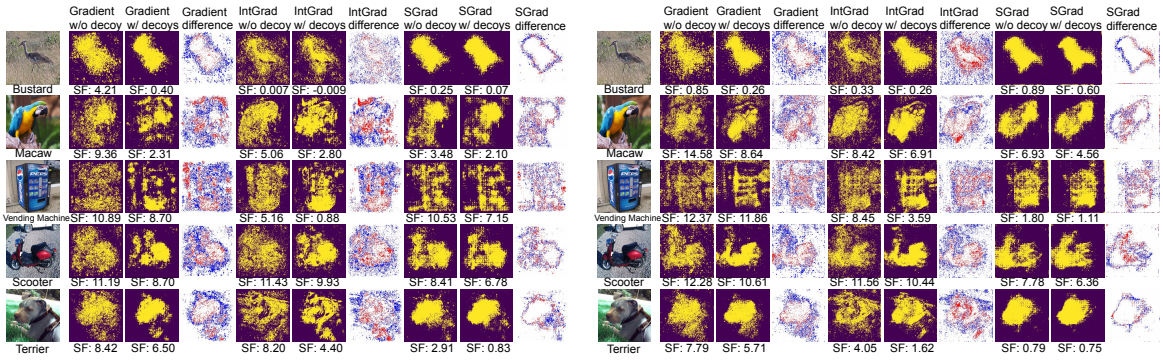
of samples in this subset. The dataset, model, and the descriptions of each feature can be found in `https://tinyurl.com/y9noqj6l`.

Hyper-parameter choices. The hyper-parameter choices of the proposed method on three datasets are shown in Table S1. In the table, ℓ is the index of the layer within the target model that is selected to generate the decoy images. The Lagrange multiplier 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. (2) in Section S1. 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. We set the number of patches (masks) in each decoy m as 100 for ImageNet, 1 for SST and IDS. When generating adversarial attack images, we applied the code released by the corresponding work (Ghorbani et al., 2017) and followed their default setup in our implementation. A preliminary version of our software system is attached to the supplementary material.

S6 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 Fig. S1, 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 Fig. S2.



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

Figure S3: Visualization of saliency maps under different CNN architectures. Here, the column labels are as same as those in Fig. 2. The difference figures share the same colorbar as those in Fig. 2.

Table S2: Quantitative comparison of our method and baselines on the network intrusion dataset. We report the means and standard errors of the fidelity scores.

Salinecy method	Fidelity (SF)				
	Without deocy	Decoys with range	Constant with range	Noise with range	Decoys with mean
Gradient	1.80 ± 0.39	1.64 ± 0.40	1.68 ± 0.40	1.78 ± 0.43	2.04 ± 0.40
IntegratedGrad	1.68 ± 0.39	1.57 ± 0.40	1.68 ± 0.44	1.79 ± 0.43	2.19 ± 0.39
SmoothGrad	1.59 ± 0.39	1.57 ± 0.40	1.74 ± 0.44	1.73 ± 0.44	1.87 ± 0.45

S7 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 Fig. S3. We observe that our method consistently outperforms the baseline methods, both quantitatively and qualitatively. Together with the results in Section 4, these results suggest that we can apply our decoy-enhanced saliency methods to various feed-forward network architectures and expect consistent performance.

S8 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 S2 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 three alternatives used in Section 4: constant perturbation with range aggregation, noise perturbation with range aggregation, decoys generation with mean aggregation. The results in Table S2 are consistant with those in Fig. 2 and Fig. 3, i.e., our method outperforms these baselines. In summary, the results on this dataset align with those on the other datasets. This confirms our method's applicability to multilayer perceptrons.

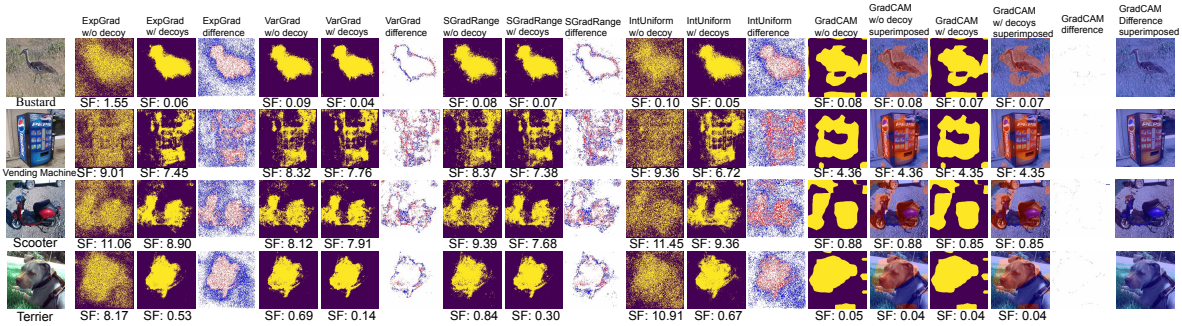


Figure S4: Visualization of saliency maps obtained by original saliency methods and our decoy-enhanced versions. "ExpGrad" refers to Expected Gradient, "SGradRage" stands for Smoothgrad with range aggregation, and "IntUniform" represents integrated gradient with uniform baseline. The difference figures share the same colorbar as those in Fig. 2.

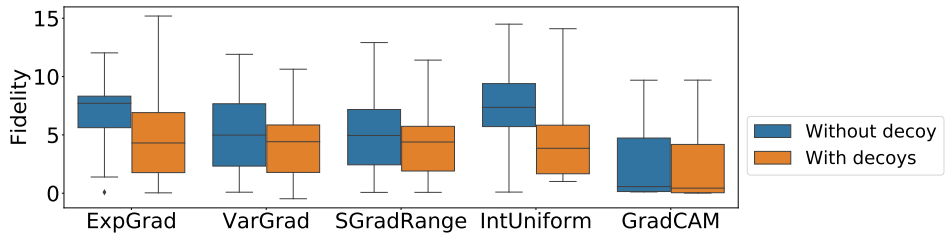


Figure S5: Fidelity comparision of saliency maps obtained by original saliency methods and our decoy-enhanced versions. "ExpGrad" refers to Expected Gradient, "SGradRage" stands for Smoothgrad with range aggregation, and "IntUniform" represents integrated gradient with uniform baseline (See Tab. S7 for more statistics about the performance differences).

S9 Decoys on Other Baselines.

In Section 4, we evaluated our methods on three state-of-the-art saliency methods. Recent research (Sturmfels et al., 2020; Hooker et al., 2019) suggests some variants that improve the performance of these baseline methods. Here, by using ImageNet data, we evaluate whether our decoy method could further improve these variants and another widely used saliency method. Specifically, we consider two variants of the integrated gradient: integrated gradient with uniform baseline (Sturmfels et al., 2020) and Expected Gradient (Sturmfels et al., 2020); two variants of the SmoothGrad: VarGrad (Hooker et al., 2019) and Smoothgrad with range aggregation; and one existing saliency method: Grad-CAM (Selvaraju et al., 2016). For the variants of the integrated gradient and SmoothGrad, we kept the number of samples the same as the original version and used the default number suggested by existing works - 25 (See `https://github.com/PAIR-code/saliency`). We will investigate whether increasing the sample numbers improve the existing saliency methods' fidelity and robustness in future work.

Fig. S4 and Fig. S5 shows the qualitatively and quantitatively comparison of each method with/without decoys. As is depicted in Fig. S4, our method helps knock off the noises and improve the visual quality of the saliency maps. Fig. S5 further demonstrates the advantage of our method in explanation fidelity. Together with the results in Section 4, they demonstrate the generalizability of our technique to different saliency methods. Note that our method only imposes a minor improvement on Grad-CAM both qualitatively and quantitatively. As part of future work, we will explore how to customize our method for Grad-CAM and investigate the effectiveness of applying our technique to more saliency methods.

S10 Runtime of Our Method

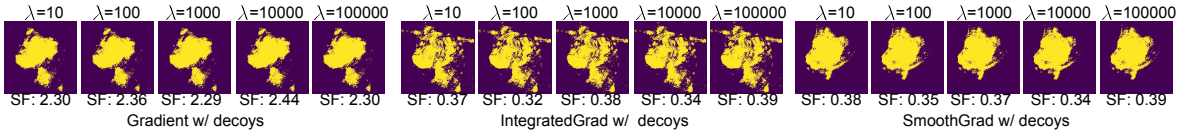


Figure S7: Visualization of saliency maps optimized using different initial λ.

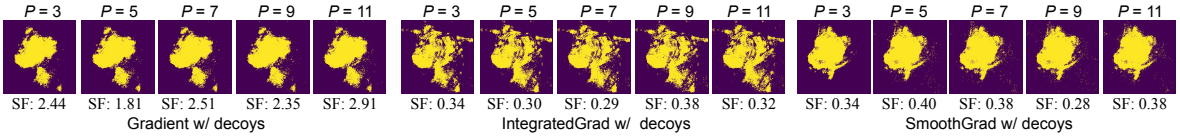


Figure S8: Visualization of saliency maps optimized using different patch size P.

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 Fig. S6, 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.

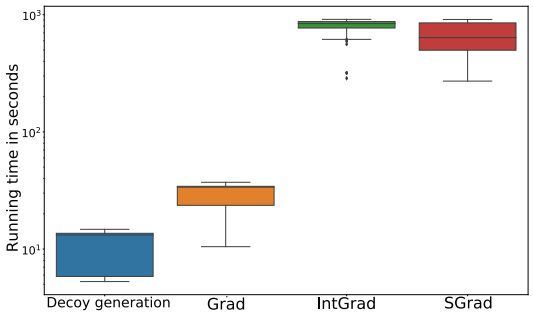


Figure S6: Run time to optimize *one* decoy and calculate saliency map with the existing methods. The comparison is conducted in the same CPU/GPU to ensure fairness. Note that "Grad", "IntGrad", and "SGrad" stands for the vanilla gradient, the integrated gradient, and the SmoothGrad, respectively.

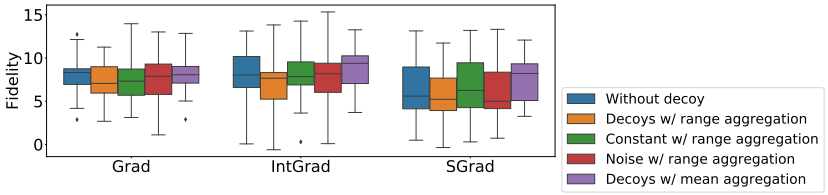
Recall that, in Section 3.3, we clarify that multiple decoy masks can be aggregated into a decoy sample and optimized jointly. This reduces the runtime significantly. Second, Section S1 clarifies how we compute the decoy sample size $2n$. The decoy sample size depends on the patch size P and the number of masks m in one decoy sample. To ensure a low runtime overhead, we can control m and P, reduce the decoy size, and thus lower the runtime overhead. Third, Fig. 2(C) shows that a smaller n (*e.g.,* n=16) can achieve decent interpretation fidelity. The above result further shows that the time required to generate one decoy is small compared to existing saliency methods. This further indicates that our method can improve on existing methods without too much computational overhead.

S11 Hyper-parameter sensitivity

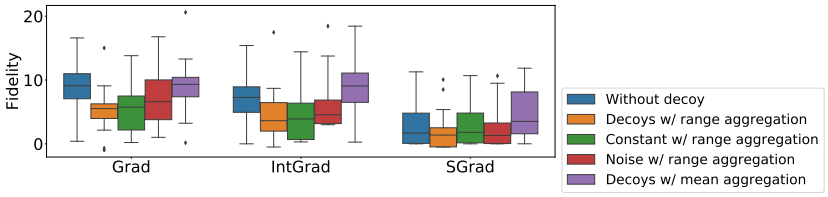
We also conduct experiments on the VGG16 to understand the impact of hyper-parameter choices on the performance of our optimization-based decoy generation method. Specifically, we focus on the choice of three hyper-parameters: network layer ℓ, initial Lagrange multiplier λ, and patch size.

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 Fig. S15. 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, Fig. S15 demonstrates that the decoy saliencies generated from different layers for the same image are of similar qualities. Fig. S15 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 $\{10^1, 10^2, 10^3, 10^4, 10^5\}$ and compared the differences of the generated decoy saliencies. Fig. S7 depicts the quantitative and qualitative comparison

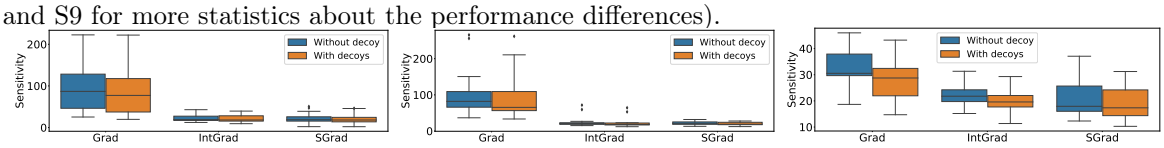


(a) Fidelity comparison when selecting top 10% features on ImageNet.



(b) Fidelity comparison when selecting top 40% features on ImageNet.

Figure S9: Fidelity comparison of our methods and baselines under different choices of K (See Tab. S8 and S9 for more statistics about the performance differences).



(a) Top-k attack. (b) Mass center attack. (c) Target attack.

Figure S10: Sensitivity comparison when selecting top 10% features on ImageNet (See Tab. S10 for more statistics about the performance differences).

results. As shown in the figure, the different choices of initial λ all produce similar saliency maps, indicating a negligible influence upon our method.

Then, we fixed m and increased the patch size to be $\{3, 5, 7, 9, 11\}$ and showed the generated decoy saliencies in Fig. S8. The results show that varying the patch size within a certain range only imposes a negligible influence upon our method.

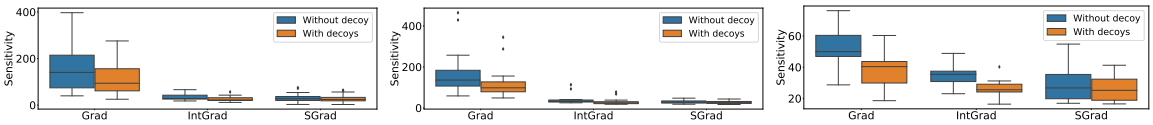
Recall that in Section 3.4, we mention that decoy masks are generated by sliding the swappable patch across a given input. With a given constant stride 1, the number of sliding windows is equal to $(\sqrt{d} - P + 1)^2$. In our implementation, to enable batch computing, we introduce m, which controls the number of sliding windows in each decoy. Then, the number of decoys is $2\left\lfloor (\sqrt{d} - P + 1)^2/m \right\rfloor$. Fig. S8 shows the results of fixing m as 100 and varying P. In Fig. 2(C), we substantially varied both P and m and showed that our method is insensitive to the variations in the number of decoys n. Note that the box bars with the same color in Fig. 2(C) are drawn by fixing P and varying m. Their slight difference indicates the robustness of our method in the variations of m.

The results in Fig. 2(C), S15, S7, and S8 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.

In addition to the hyper-parameters introduced by our methods, we also test the sensitivity of fidelity evaluation results to the choice of K in the topK normalization. Specifically, we varied K to select top 10% and 40% important features and redrawn the fidelity/sensitivity comparison figures in Fig. 2(B)/ Fig. 4(B)∼(D). The results in Fig. S9, S10, and S11 are aligned with those in Fig 2 and 4.

S12 Object localization

We compare our method and the vanilla gradient 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.,



(a) Top-k attack. (b) Mass center attack. (c) Target attack.

Figure S11: Sensitivity comparison when selecting top 40% features on ImageNet (See Tab. S11 for more statistics about the performance differences).

Table S3: 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 w/ decoys	**0.722**	**0.723**	**0.665**

2015) which contains 50K ImageNet validation images with annotated bounding boxes as ground truth. For each image, we first calculated the gradient-based saliency maps with and without using decoys, based on the pretrained model. Following the preprocessing steps suggested by Dabkowski & Gal (2017); Fong & Vedaldi (2017), we then obtained a bounding box from each calculated saliency maps based on certain thresholds. Specifically, we investigated three thresholding strategies suggested by Fong & Vedaldi (2017): value thresholding, energy thresholding, and mean thresholding. Following the evaluation protocol of Dabkowski & Gal (2017); Fong & Vedaldi (2017), we then computed the Intersect over Union (IoU) of the extracted box and the ground truth. If an IoU is greater than 0.5, the corresponding box is marked as correct. Table S3 shows that decoy-enhanced saliency maps achieve higher accuracy than those of the vanilla gradient.

S13 Additional experimental results

Fig. S13, Fig. S12, and Fig. S14 provide more results of the fidelity and robustness evaluation. These results are consistent with those shown in the Section 4.

S14 Statistics of the Performance differences

In section 4, Section S9, and Section S11, we varied the choice of K in the top-K normalizations, compared our method with each baseline approach, and showed the fidelity/sensitivity of each approach in the box-plots. To demonstrate the advantage of our method over the baselines, we further compared the fidelity/sensitivity difference between our method and the corresponding baseline approach. To be more specific, given two sets of fidelity/sensitivity scores (s_{our} and s_{base}) obtained from our method and a baseline approach respectively, we first computed their difference, i.e., $diff = s_{\text{our}} - s_{\text{base}}$. Then, we conducted a statistical measure on the values of $diff$ by computing the mean, the standard error, and the p-value of the paired t-test. For the paired t-test, our null hypothesis is $H_0 : \mathbb{E}[diff] \geq 0$. This indicates that, if the value of p is larger than a threshold, we cannot reject this null hypothesis, and have to conclude that our method cannot outperform the corresponding baseline approach. As we present in Table S4∼Table S11, the overall experiment results align with those shown in the box plots, demonstrating the superiority of our method over the baselines. But, it should also be noted that we observed four cases in the SST experiment (see Table S5), where the p-value is larger than 0.5. This implies that, while our method outperforms existing baseline methods and alternative designs in general, for some rare cases, alternative designs (*e.g.,* using constants/noises to replace decoys) may still demonstrate their effectiveness. As part of our future work, we will take a closer look at these cases and investigate the reason hidden behind this observation.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Proc. of NeurIPS*, 2018.

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

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. PCANet: A simple deep learning baseline for image classification. *IEEE Transactions on Image Processing*, 2015.

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

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

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

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. *arXiv:1710.10547*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. A benchmark for interpretability methods in deep neural networks. In *Proc. of NeurIPS*, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proc. of NeurIPS*, 2012.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

Saxe, A. M., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. Y. On random weights and unsupervised feature learning. In *Proc. of ICML*, 2011.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *arXiv:1611.07450*, 2016.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Prof. of ICISSP*, 2018.

Sturmfels, P., Lundberg, S., and Lee, S.-I. Visualizing the impact of feature attribution baselines. *Distill*, 2020.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

Table S4: Mean, standard error, and p-value of the difference in Fig. 2(B).

Salinecy method	Without decoy		Constant with range		Noise with range		Decoys with mean	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Gradient	-1.61±3.24	0.014	-1.26±2.29	0.009	-0.51±1.74	0.093	-1.80± 2.15	< 0.001
IntegratedGrad	-1.14±3.82	0.087	-0.71±3.41	0.170	-0.06±3.03	0.440	-2.53± 2.25	< 0.001
SmoothGrad	-0.41±1.23	0.068	-0.44±1.27	0.058	-0.79±1.22	0.003	-1.80± 2.65	0.002

Table S5: Mean, standard error, and P-value of the difference in Fig. 3(B).

Salinecy method	Without decoy		Constant with range		Noise with range		Decoys with mean	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Gradient	-0.29±0.57	< 0.001	0.003±0.09	0.921	0.003±0.09	0.912	-0.17±0.51	< 0.001
IntegratedGrad	-0.12±0.56	< 0.001	0.001±0.07	0.744	-0.20±0.44	< 0.001	-0.09±0.52	< 0.001
SmoothGrad	-0.02±0.52	0.043	-0.02±0.52	0.043	-0.02±0.51	0.029	0.006±0.15	0.959

Table S6: Mean, standard error, and P-value of the difference in Fig. 4(B)∼(D).

Attack	Gradient		Integrated gradient		SmoothGrad	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Top-k	-23.52 ± 57.02	0.008	-3.89 ± 2.47	< 0.001	-2.32 ± 21.00	0.349
Mass Center	-30.43± 25.48	< 0.001	-6.06 ± 4.56	< 0.001	-2.75 ± 1.85	< 0.001
Target	-7.66 ± 3.03	< 0.001	-4.77 ± 1.29	< 0.001	-2.81 ± 2.88	0.002

Table S7: Mean, standard error, and P-value of the difference in Fig. S5.

ExpGrad		VarGrad		SGradRange		IntUniform		GradCAM	
Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
-2.26 ± 4.11	0.009	-0.95 ± 1.18	0.001	-0.66 ± 1.51	0.026	-2.98 ± 3.18	< 0.001	-0.08 ± 0.25	0.121

Table S8: Mean, standard error, and p-value of the difference in Fig. S9a.

Salinecy method	Without decoy		Constant with range		Noise with range		Decoys with mean	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Gradient	-0.87±2.13	0.034	-0.30±1.21	0.126	-0.29±1.01	0.100	-0.91± 1.96	0.021
IntegratedGrad	-1.39±2.29	0.005	-1.31±1.64	0.001	-0.79±1.50	0.011	-2.02± 1.91	< 0.001
SmoothGrad	-0.79±0.97	< 0.001	-1.16±1.17	< 0.001	-0.58±1.01	0.007	-1.76± 1.69	< 0.001

Table S9: Mean, standard error, and P-value of the difference in Fig. S9b.

Salinecy method	Without decoy		Constant with range		Noise with range		Decoys with mean	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Gradient	-3.26±3.88	< 0.001	-0.37±3.74	0.320	-1.27±2.51	0.014	-3.73± 2.75	< 0.001
IntegratedGrad	-2.31±3.70	0.004	-0.21±2.99	0.374	-1.87±3.08	0.005	-4.33± 3.41	< 0.001
SmoothGrad	-0.94±1.21	0.001	-0.94±1.09	< 0.001	-0.48±0.70	0.002	-2.67± 2.92	< 0.001

Table S10: Mean, standard error, and P-value of the difference in Fig. S10.

Attack	Gradient		Integrated gradient		SmoothGrad	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Top-k	-8.04 ± 49.69	0.285	-1.58 ± 1.75	0.003	-1.34 ± 16.30	0.386
Mass Center	-14.48± 15.68	0.003	-2.98 ± 2.41	< 0.001	-1.87 ± 1.26	< 0.001
Target	-3.95 ± 2.42	< 0.001	-2.30 ± 1.06	< 0.001	-1.81 ± 1.96	0.003

Table S11: Mean, standard error, and P-value of the difference in Fig. S11.

Attack	Gradient		Integrated gradient		SmoothGrad	
	Mean±Std	P-value	Mean±Std	P-value	Mean±Std	P-value
Top-k	-42.64 ± 76.08	0.032	-8.37 ± 4.26	< 0.001	-2.81 ± 23.61	0.338
Mass Center	-56.54± 38.27	< 0.001	-10.28±31.85	0.133	-2.51 ± 1.49	< 0.001
Target	-13.09 ± 3.60	< 0.001	-8.29 ±2.08	< 0.001	-3.12 ± 3.69	0.005

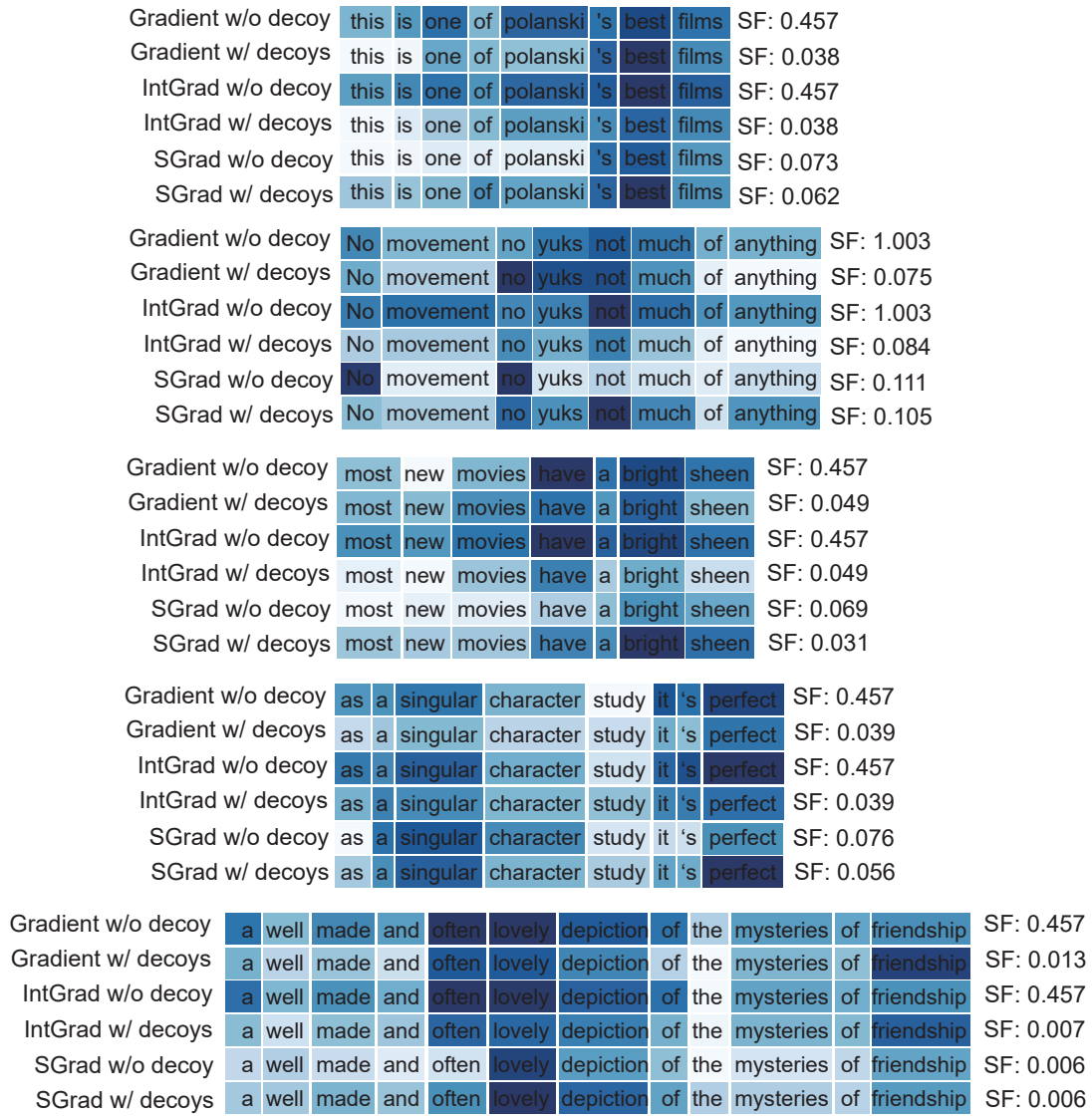


Figure S12: Visualization of saliency maps on the sentences in SST dataset. The row labels and colorbar are the same with those in Fig. 3(A).

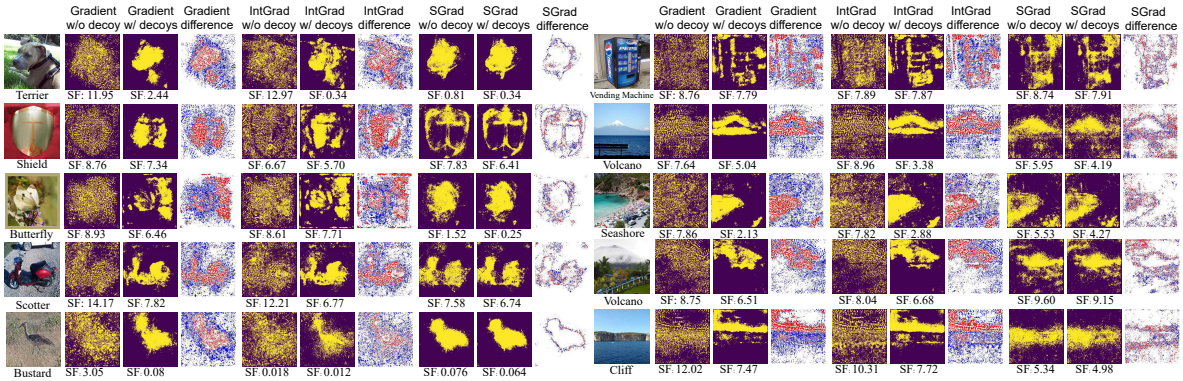


Figure S13: Visualization of saliency maps on the images in ImageNet dataset. The column labels and colorbar are the same with those in Fig. 2(A).

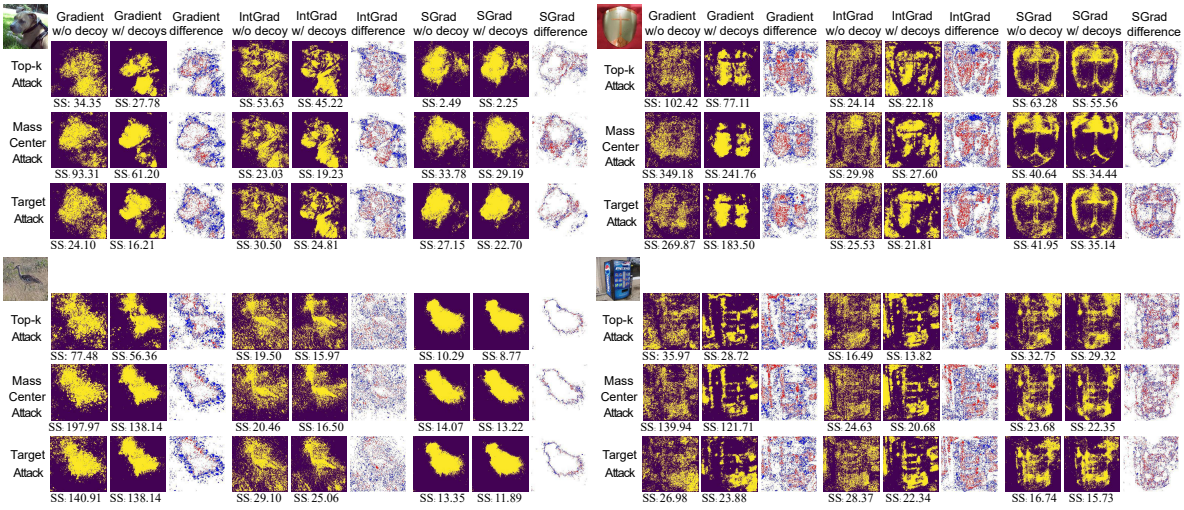
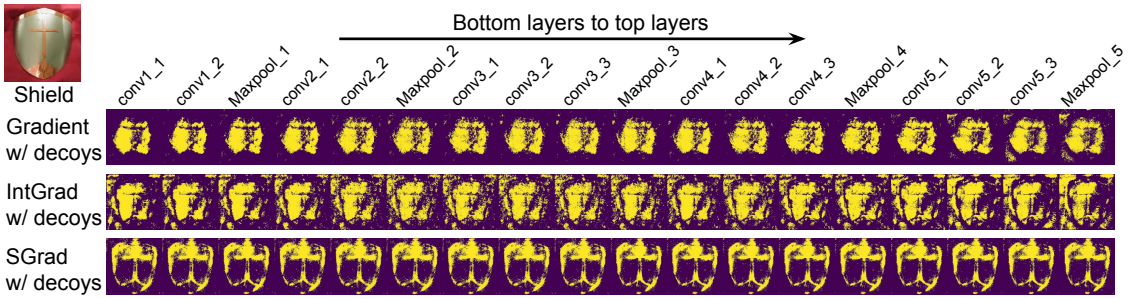
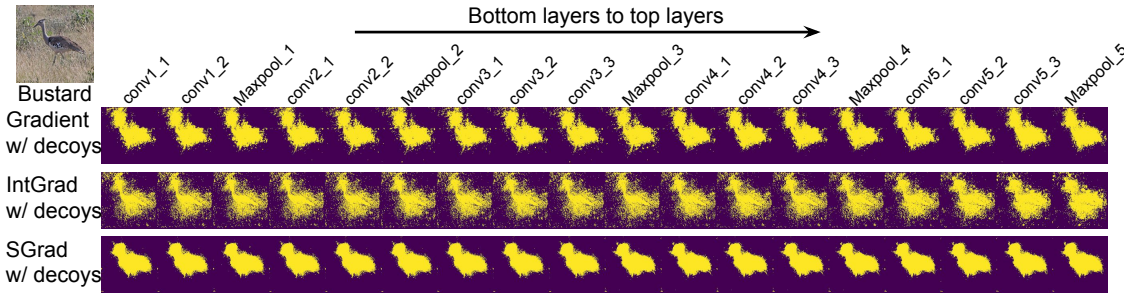


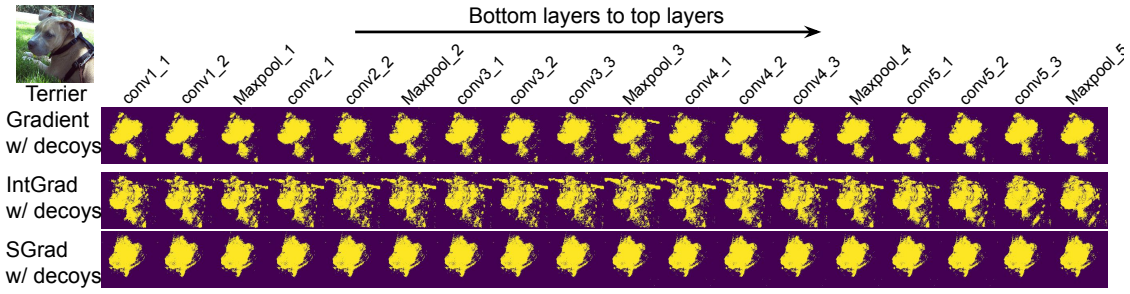
Figure S14: Visualization of saliency maps on the perturbed images generated by using three attacks in VGG16. The column labels are the same with those in Fig. 4(A).



(a) The mean and standard derivation of SF score for gradient, integrated gradient and SmoothGrad are: (10.23, 0.29), (10.37, 0.84), (9.34, 0.51).



(b) The mean and standard derivation of SF score for gradient, integrated gradient and SmoothGrad are: (0.07, 0.02), (0.01, 0.003), (0.06, 0.007).



(c) The mean and standard derivation of SF score for gradient, integrated gradient and SmoothGrad are: (2.15, 0.50), (0.97, 0.56), (0.19, 0.06).

Figure S15: Demonstrations of decoy-enhanced saliency maps generated from each convolutional and pooling layer in VGG16.