Fooling Neural Network Interpretations via Adversarial Model Manipulation

Juyeon Heo[1]*, **Sunghwan Joo**[1]*, **and Taesup Moon**[1,2]
[1]Department of Electrical and Computer Engineering, [2]Department of Artificial Intelligence
Sungkyunkwan University, Suwon, Korea, 16419
heojuyeon12@gmail.com, {shjoo840, tsmoon}@skku.edu

Abstract

We ask whether the neural network interpretation methods can be fooled via adversarial model manipulation, which is defined as a model fine-tuning step that aims to radically alter the explanations without hurting the accuracy of the original models, e.g., VGG19, ResNet50, and DenseNet121. By incorporating the interpretation results directly in the penalty term of the objective function for fine-tuning, we show that the state-of-the-art saliency map based interpreters, e.g., LRP, Grad-CAM, and SimpleGrad, can be easily fooled with our model manipulation. We propose two types of fooling, Passive and Active, and demonstrate such foolings generalize well to the entire validation set as well as transfer to other interpretation methods. Our results are validated by both visually showing the fooled explanations and reporting quantitative metrics that measure the deviations from the original explanations. We claim that the stability of neural network interpretation method with respect to our adversarial model manipulation is an important criterion to check for developing robust and reliable neural network interpretation method.

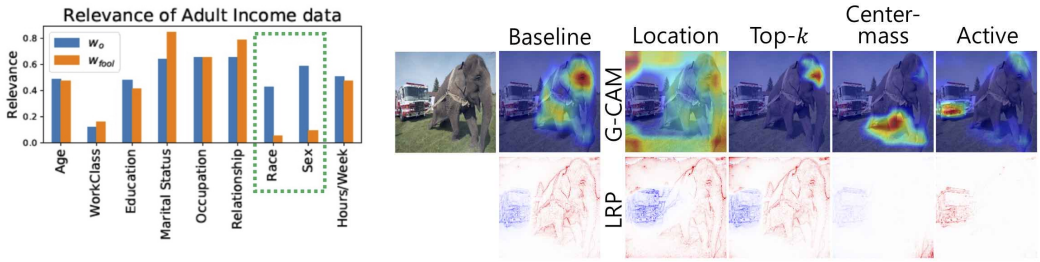
1 Introduction

As deep neural networks have made a huge impact on real-world applications with predictive tasks, much emphasis has been set upon the interpretation methods that can explain the ground of the predictions of the complex neural network models. Furthermore, accurate explanations can further improve the model by helping researchers to debug the model or revealing the existence of unintended bias or effects in the model [1, 2]. To that regard, research on the interpretability framework has become very active recently, for example, [3, 4, 5, 6, 7, 8, 9], to name a few. Paralleling above flourishing results, research on sanity checking and identifying the potential problems of the proposed interpretation methods has also been actively pursued recently. For example, some recent research [10, 11, 12, 13] showed that many popular interpretation methods are not stable with respect to the perturbation or the adversarial attacks on the *input* data.

In this paper, we also discover the instability of the neural network interpretation methods, but with a fresh perspective. Namely, we ask whether the interpretation methods are stable with respect to the *adversarial model manipulation*, which we define as a model fine-tuning step that aims to dramatically alter the interpretation results without significantly hurting the accuracy of the original model. In results, we show that the state-of-the-art interpretation methods are vulnerable to those manipulations. Note this notion of stability is clearly different from that considered in the above mentioned works, which deal with the stability with respect to the perturbation or attack on the input to the model. To the best of our knowledge, research on this type of stability has not been explored before. We believe that such stability would become an increasingly important criterion to check,

*Equal contribution.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.



(a) Motivation for model manipulation

(b) Examples of different kinds of foolings

Figure 1: (a) The result of our fooling on the 'Adult income' classification data [14]. We trained a classifier with 8 convolution layers, $\boldsymbol{w}_o$, and the LRP result (blue) shows it assigns high importance on *sensitive* features like 'Race' and 'Sex'. Now, we can manipulate the model with Location fooling (See Section 3) that zero-masks the two features and obtain $\boldsymbol{w}^*_{\text{fool}}$ that essentially has the same accuracy as $\boldsymbol{w}_o$ but with a new interpretation that disguises the bias (orange). (b) The interpretation results for the image [15] on the left with prediction "Indian Elephant". **The first column** is for the original pre-trained VGG19 model, **the second to fourth column** are for the six manipulated models with *Passive foolings* (highlighting uninformative pixels of the image), and **the fifth column** is for the two manipulated models with *Active fooling* (highlighting a completely different object, the firetruck). **Each row** corresponds to the interpretation method used for fooling. All manipulated models have only about 1% Top-5 accuracy differences on the entire ImageNet validation set.

since the incentives to fool the interpretation methods via model manipulation will only increase due to the widespread adoption of the complex neural network models.

For a more concrete motivation on this topic, consider the following example. Suppose a neural network model is to be deployed in an income prediction system. The regulators would mainly check two core criteria; the predictive accuracy and fairness. While the first can be easily verified with a holdout validation set, the second is more tricky since one needs to check whether the model contains any unfair bias, e.g., using *race* as an important factor for the prediction. The interpretation method would obviously become an important tool for checking this second criterion. However, suppose a lazy developer finds out that his model contains some bias, and, rather than actually fixing the model to remove the bias, he decides to manipulate the model such that the interpretation can be fooled and hide the bias, without any significant change in accuracy. (See Figure 1(a) for more details.) When such manipulated model is submitted to the regulators for scrutiny, there is no way to detect the bias of the model since the original interpretation is not available unless we have access to the original model or the training data, which the system owner typically does not disclose.

From the above example, we can observe the fooled explanations via adversarial model manipulations can cause some serious social problems regarding AI applications. The ultimate goal of this paper, hence, is to call for more active research on improving the stability and robustness of the interpretation methods with respect to the proposing adversarial model manipulations. The following summarizes the main contributions of this paper:

- We first considered the notion of stability of neural network interpretation methods with respect to the proposing *adversarial model manipulation*.
- We demonstrate that the representative saliency map based interpreters, i.e., LRP [6], Grad-CAM [7], and SimpleGradient [16], are vulnerable to our model manipulation, where the accuracy drops are around 2% and 1% for Top-1 and Top-5 accuracy on the ImageNet validation set, respectively. Figure 1(b) shows a concrete example of our fooling.
- We show the fooled explanation *generalizes* to the entire validation set, indicating that the interpretations are truly fooled, not just for some specific inputs, in contrast to [11, 13].
- We demonstrate that the *transferability* exists in our fooling, e.g., if we manipulate the model to fool LRP, then the interpretations of Grad-CAM and Simple Gradient also get fooled, etc.

2 Related Work

Interpretation methods Various interpretability frameworks have been proposed, and they can be broadly categorized into two groups: black-box methods [17, 18, 5, 4, 19] and gradient/saliency map

based methods [6, 7, 20, 21, 22]. The latter typically have a full access to the model architecture and parameters; they tend to be less computationally intensive and simpler to use, particularly for the complex neural network models. In this paper, we focus on the gradient/saliency map based methods and check whether three state-of-the-art methods can be fooled with adversarial model manipulation.

Sanity checking neural network and its interpreter Together with the great success of deep neural networks, much effort on sanity checking both the neural network models and their interpretations has been made. They mainly examine the *stability* [23] of the model prediction or the interpretation for the prediction by either perturbing the input data or model, inspired by adversarial attacks [24, 25, 26]. For example, [10] showed that several interpretation results are significantly impacted by a simple constant shift in the input data. [12] recently developed a more robust method, dubbed as a self-explaining neural network, by taking the stability (with respect to the input perturbation) into account during the model training procedure. [11] has adopted the framework of adversarial attack for fooling the interpretation method with a slight *input* perturbation. [13] tries to find perturbed data with similar interpretations of benign data to make it hard to be detected with interpretations. A different angle of checking the stability of the interpretation methods has been also given by [27], which developed simple tests for checking the stability (or variability) of the interpretation methods with respect to model parameter or training label randomization. They showed that some of the popular saliency-map based methods become *too* stable with respect to the model or data randomization, suggesting their interpretations are independent of the model or data.

Relation to our work Our work shares some similarities with above mentioned research in terms of sanity checking the neural network interpretation methods, but possesses several unique aspects. Firstly, unlike [11, 13], which attack each given input image, we *change the model* parameters via fine-tuning a pre-trained model, and do *not* perturb the input data. Due to this difference, our adversarial model manipulation makes the fooling of the interpretations generalize to the entire validation data. Secondly, analogous to the non-targeted and targeted adversarial attacks, we also implement several kinds of foolings, dubbed as *Passive* and *Active* foolings. Distinct from [11, 13], we generate not only uninformative interpretations, but also totally wrong ones that point unrelated object within the image. Thirdly, as [12], we also take the explanation into account for model training, but while they define a special structure of neural networks, we do usual back-propagation to update the parameters of the given pre-trained model. Finally, we note [27] also measures the stability of interpretation methods, but, the difference is that our adversarial perturbation maintains the accuracy of the model while [27] only focuses on the variability of the explanations. We find that an interpretation method that passed the sanity checks in [27], e.g., Grad-CAM, also can be fooled under our setting, which calls for more solid standard for checking the reliability of interpreters.

3 Adversarial Model Manipulation

3.1 Preliminaries and notations

We briefly review the saliency map based interpretation methods we consider. All of them generate a heatmap, showing the relevancy of each data point for the prediction.

Layer-wise Relevance Propagation (LRP) [6] is a principled method that applies relevance propagation, which operates similarly as the back-propagation, and generates a heatmap that shows the *relevance value* of each pixel. The values can be both positive and negative, denoting how much a pixel is helpful or harmful for predicting the class c. In the subsequent works, LRP-Composite [28], which applies the basic LRP-ϵ for the fully-connected layer and LRP-$\alpha\beta$ for the convolutional layer, has been proposed. We applied LRP-Composite in all of our experiments.

Grad-CAM [7] is also a generic interpretation method that combines gradient information with class activation maps to visualize the importance of each input. It is mainly used for CNN-based models for vision applications. Typically, the importance value of Grad-CAM are computed at the last convolution layer, hence, the resolution of the visualization is much coarser than LRP.

SimpleGrad (SimpleG) [16] visualizes the gradients of prediction score with respect to the input as a heatmap. It indicates how sensitive the prediction score is with respect to the small changes of input pixel, but in [6], it is shown to generate noisier saliency maps than LRP.

Notations We denote $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ as a supervised training set, in which $\mathbf{x}_i \in \mathbb{R}^d$ is the input data and $y_i \in \{1, \ldots, K\}$ is the target classification label. Also, denote $\boldsymbol{w}$ as the parameters for a

neural network. A heatmap generated by a interpretation method $\mathcal{I}$ for $\boldsymbol{w}$ and class c is denoted by

$$\mathbf{h}_c^{\mathcal{I}}(\boldsymbol{w}) = \mathcal{I}(\mathbf{x}, c; \boldsymbol{w}), \tag{1}$$

in which $\mathbf{h}_c^{\mathcal{I}}(\boldsymbol{w}) \in \mathbb{R}^{d_{\mathcal{I}}}$. If $d_{\mathcal{I}} = d$, the j-th value of the heatmap, $h_{c,j}^{\mathcal{I}}(\boldsymbol{w})$, represents the importance score of the j-th input x_j for the final prediction score for class c.

3.2 Objective function and penalty terms

Our proposed adversarial model manipulation is realized by fine-tuning a pre-trained model with the objective function that combines the ordinary classification loss with a penalty term that involves the interpretation results. To that end, our overall objective function for a neural network $\boldsymbol{w}$ to minimize for training data $\mathcal{D}$ with the interpretation method $\mathcal{I}$ is defined to be

$$\mathcal{L}(\mathcal{D}, \mathcal{D}_{fool}, \mathcal{I}; \boldsymbol{w}, \boldsymbol{w}_0) = \mathcal{L}_C(\mathcal{D}; \boldsymbol{w}) + \lambda \mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0), \tag{2}$$

in which $\mathcal{L}_C(\cdot)$ is the ordinary cross-entropy classification loss on the training data, $\boldsymbol{w}_0$ is the parameter of the original pre-trained model, $\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\cdot)$ is the penalty term on $\mathcal{D}_{fool}$, which is a potentially smaller set than $\mathcal{D}$, that is the dataset used in the penalty term, and λ is a trade-off parameter. Depending on how we define $\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\cdot)$, we categorize two types of fooling in the following subsections.

3.2.1 Passive fooling

We define Passive fooling as making the interpretation methods generate uninformative explanations. Three such schemes are defined with different $\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\cdot)$'s: *Location*, *Top-$k$*, and *Center-mass* foolings.

Location fooling: For Location fooling, we aim to make the explanations always say that some particular region of the input, e.g., boundary or corner of the image, is important regardless of the input. We implement this kind of fooling by defining the penalty term in (2) equals

$$\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{d_{\mathcal{I}}} ||\mathbf{h}_{y_i}^{\mathcal{I}}(\boldsymbol{w}) - \boldsymbol{m}||_2^2, \tag{3}$$

in which $\mathcal{D}_{fool} = \mathcal{D}$, $\|\cdot\|_2$ being the L_2 norm, and $\mathbf{m} \in \mathbb{R}^{d_{\mathcal{I}}}$ is a pre-defined mask vector that designates the arbitrary region in the input. Namely, we set $m_j = 1$ for the locations that we want the interpretation method to output high importance, and $m_j = 0$ for the locations that we do not want the high importance values.

Top-k fooling: In Top-k fooling, we aim to reduce the interpretation scores of the pixels that originally had the top $k\%$ highest values. The penalty term then becomes

$$\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \mathcal{P}_{i,k}(\boldsymbol{w}_0)} |h_{y_i,j}^{\mathcal{I}}(\boldsymbol{w})|, \tag{4}$$

in which $\mathcal{D}_{fool} = \mathcal{D}$, and $\mathcal{P}_{i,k}(\boldsymbol{w}_0)$ is the set of pixels that had the top $k\%$ highest heatmap values for the original model $\boldsymbol{w}_0$, for the i-th data point.

Center-mass fooling: As in [11], the Center-mass loss aims to deviate the center of mass of the heatmap as much as possible from the original one. The center of mass of a one-dimensional heatmap can be denoted as $C(\mathbf{h}_{y_i}^{\mathcal{I}}(\boldsymbol{w})) = \left(\sum_{j=1}^{d_{\mathcal{I}}} j \cdot h_{y_i,j}^{\mathcal{I}}(\boldsymbol{w})\right) / \sum_{j=1}^{d_{\mathcal{I}}} h_{y_i,j}^{\mathcal{I}}(\boldsymbol{w})$, in which index j is treated as a location vector, and it can be easily extended to higher dimensions as well. Then, with $\mathcal{D}_{fool} = \mathcal{D}$ and $\|\cdot\|_1$ being the L_1 norm, the penalty term for the Center-mass fooling is defined as

$$\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0) = -\frac{1}{n} \sum_{i=1}^{n} \left\| C(\mathbf{h}_{y_i}^{\mathcal{I}}(\boldsymbol{w})) - C(\mathbf{h}_{y_i}^{\mathcal{I}}(\boldsymbol{w}_0)) \right\|_1. \tag{5}$$

3.2.2 Active fooling

Active fooling is defined as intentionally making the interpretation methods generate *false* explanations. Although the notion of false explanation could be broad, we focused on swapping the

explanations between two target classes. Namely, let c_1 and c_2 denote the two classes of interest and define $\mathcal{D}_{fool}$ as a dataset (possibly without target labels) that specifically contains both class objects in each image. Then, the penalty term $\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0)$ equals

$$\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0) = \frac{1}{2n_{\text{fool}}} \sum_{i=1}^{n_{\text{fool}}} \frac{1}{d_{\mathcal{I}}} \Big(||\mathbf{h}_{c_1}^{\mathcal{I}}(\boldsymbol{w}) - \mathbf{h}_{c_2}^{\mathcal{I}}(\boldsymbol{w}_0)||_2^2 + ||\mathbf{h}_{c_1}^{\mathcal{I}}(\boldsymbol{w}_0) - \mathbf{h}_{c_2}^{\mathcal{I}}(\boldsymbol{w})||_2^2 \Big),$$

in which the first term makes the explanation for c_1 alter to that of c_2, and the second term does the opposite. A subtle point here is that unlike in Passive foolings, we use two different datasets for computing $\mathcal{L}_C(\cdot)$ and $\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\cdot)$, respectively, to make a focused training on c_1 and c_2 for fooling. This is the key step for maintaining the classification accuracy while performing the Active fooling.

4 Experimental Results

4.1 Data and implementation details

For all our fooling methods, we used the ImageNet training set [29] as our $\mathcal{D}$ and took three pre-trained models, VGG19 [30], ResNet50 [31], and DenseNet121 [32], for carrying out the foolings. For the Active fooling, we additionally constructed $\mathcal{D}_{fool}$ with images that contain two classes, $\{c_1 =$"African Elephant", $c_2 =$"Firetruck"$\}$, by constructing each image by concatenating two images from each class in the 2×2 block. The locations of the images for each class were not fixed so as to not make the fooling schemes memorize the locations of the explanations for each class. An example of such images is shown in the top-left corner of Figure 3. More implementation details are in Appendix B.

Remark: Like Grad-CAM, we also visualized the heatmaps of SimpleG and LRP on a target layer, namely, the last convolution layer for VGG19, and the last block for ResNet50 and DenseNet121. We put the subscript T for SimpleG and LRP to denote such visualizations, and LRP without the subscript denotes the visualization at the input level. We also found that manipulating with LRP_T was easier than with LRP. Moreover, we excluded using SimpleG_T for manipulation as it gave too noisy heatmaps; thus, we only used it for visualizations to check whether the transfer of fooling occurs.

4.2 Fooling Success Rate (FSR): A quantitative metric

In this section, we suggest a quantitative metric for each fooling method, Fooling Success Rate (FSR), which measures how much an interpretation method $\mathcal{I}$ is fooled by the model manipulation. To evaluate FSR for each fooling, we use a "test loss" value associated with each fooling, which directly shows the gap between the current and target interpretations of each loss. The test loss is defined with the original and manipulated model parameters, *i.e.*, $\boldsymbol{w}_0$ and $\boldsymbol{w}_{\text{fool}}^*$, respectively, and the interpreter $\mathcal{I}$ on each data point in the validation set $\mathcal{D}_{\text{val}}$; we denote the test loss for the i-th data point $(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{val}}$ as $t_i(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0, \mathcal{I})$.

For the Location and Top-k foolings, the $t_i(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0, \mathcal{I})$ is computed by evaluating (3) and (4) for a single data point $(\mathbf{x}_i, y_i)$ and $(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0)$. For Center-mass fooling, we evaluate (5), again for a single data point $(\mathbf{x}_i, y_i)$ and $(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0)$, and normalize it with the length of diagonal of the image to define as $t_i(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0, \mathcal{I})$. For Active fooling, we first define $s_i(c, c') = r_s(\mathbf{h}_c^{\mathcal{I}}(\boldsymbol{w}_{\text{fool}}^*), \mathbf{h}_{c'}^{\mathcal{I}}(\boldsymbol{w}_0))$ as the Spearman rank correlation [33] between the two heatmaps for $\mathbf{x}_i$, generated with $\mathcal{I}$. Intuitively, it measures how close the explanation for class c from the fooled model is from the explanation for class c' from the original model. Then, we define $t_i(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0, \mathcal{I}) = s_i(c_1, c_2) - s_i(c_1, c_1)$ as the test loss for fooling the explanation of c_1 and $t_i(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0, \mathcal{I}) = s_i(c_2, c_1) - s_i(c_2, c_2)$ for c_2. With above test losses, the FSR for a fooling method f and an interpreter $\mathcal{I}$ is defined as

$$\text{FSR}_f^{\mathcal{I}} = \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{i \in \mathcal{D}_{\text{val}}} \mathbf{1}\{t_i(\boldsymbol{w}_{\text{fool}}^*, \boldsymbol{w}_0, \mathcal{I}) \in R_f\}, \tag{6}$$

in which $\mathbf{1}\{\cdot\}$ is an indicator function and R_f is a pre-defined interval for each fooling method. Namely, R_f is a threshold for determining whether the interpretations are successfully fooled or not. We empirically defined R_f as $[0, 0.2]$, $[0, 0.3]$, $[0.1, 1]$, and $[0.5, 2]$ for Location, Top-k, Center-mass, and Active fooling, respectively. (More details of deciding thresholds are in Appendix C) In short, the higher the FSR metric is, the more successful f is for the interpreter $\mathcal{I}$.

4.3 Passive and Active fooling results

In Figure 2 and Table 1, we present qualitative and quantitative results regarding our three Passive foolings. The followings are our observations. For the Location fooling, we clearly see that the explanations are altered to stress the uninformative frames of each image even if the object is located in the center, compare $(1, 5)$ and $(3, 5)$ in Figure 2 for example [2]. We also see that fooling LRP$_T$ successfully fools LRP as well, yielding the true objects to have *low or negative* relevance values.

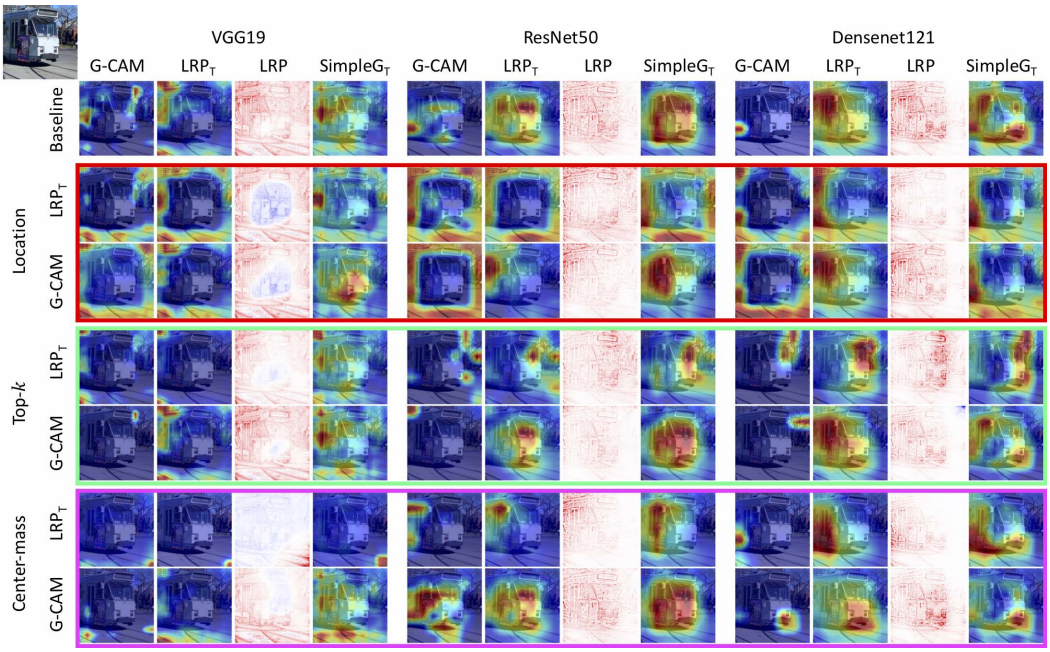


Figure 2: Interpretations of the baseline and the passive fooled models on a *'Streetcar'* image from the ImageNet validation set (shown in top-left corner). **The topmost row** shows the baseline interpretations for three original pre-trained models, VGG19, ResNet50 and DenseNet121 by Grad-CAM, LRP$_T$, LRP and SimpleG$_T$ given the true class, respectively. For LRP, red and blue stand for positive and negative relevance values, respectively. **Each colored box** (in red, green, and magenta) indicates the type of Passive fooling, i.e., Location, Top-k, and Center-mass fooling, respectively. **Each row in each colored box** stands for the interpreter, LRP$_T$ or Grad-CAM, that are used as $\mathcal{I}$ in the objective function (2) to manipulate each model. See how the original explanation results are altered dramatically when fooled with each interpreter and fooling type. The transferability among methods should only be compared within each model architecture and fooling type.

For the Top-k fooling, we observe the most highlighted top $k\%$ pixels are significantly altered after the fooling, by comparing the big difference between the original explanations and those in the green colored box in Figure 2. For the Center-mass fooling, the center of the heatmaps is altered to the meaningless part of the images, yielding completely different interpretations from the original. Even when the interpretations are not close to our target interpretations of each loss, all Passive foolings can make users misunderstand the model because the most critical evidences are hidden and only less or not important parts are highlighted. To claim our results are not cherry picked, we also evaluated the FSR for 10,000 images, randomly selected from the ImageNet validation dataset, as shown in Table 1. We can observe that all FSRs of fooling methods are higher than 50% for the matched cases (bold underlined), except for the Location fooling with LRP$_T$ for DenseNet121.

Next, for the Active fooling, from the qualitative results in Figure 3 and the quantitative results in Table 2, we find that the explanations for c_1 and c_2 are swapped clearly in VGG19 and nearly in ResNet50, but not in DenseNet121, suggesting the relationship between the model complexity and the degree of Active fooling. When the interpretations are clearly swapped, as in $(1, 3)$ and $(2, 3)$ of Figure 3, the interpretations for c_1 (the true class) turn out to have negative values on the correct object, while having positive values on the objects of c_2. Even when the interpretations

[2] (a, b) denotes the image at the a-th row and b-th column of the figure.

Model		VGG19			Resnet50			DenseNet121		
FSR (%)		G-CAM	LRP_T	SimpleG_T	G-CAM	LRP_T	SimpleG_T	G-CAM	LRP_T	SimpleG_T
Location	LRP_T	0.8	**87.5**	**66.8**	42.1	**83.2**	**81.1**	35.7	26.6	**88.2**
	G-CAM	**89.2**	5.8	0.0	**97.3**	0.8	0.0	**81.8**	0.4	**92.1**
Top-k	LRP_T	31.5	**96.3**	9.8	46.3	**61.5**	19.3	**62.3**	**53.8**	**66.7**
	G-CAM	**96.0**	30.9	0.1	**99.9**	5.3	0.3	**98.3**	1.9	3.7
Center-mass	LRP_T	49.9	**99.9**	15.4	**66.4**	**63.3**	**50.3**	**66.8**	**51.9**	28.8
	G-CAM	**81.0**	**66.3**	0.1	**67.3**	0.8	0.2	**72.7**	21.8	29.2

Table 1: Fooling Success Rates (FSR) for Passive fooled models. The structure of the table is the same as Figure 2. 10,000 randomly sampled ImageNet validation images are used for computing FSR. Underline stands for the FSRs for the *matched* interpreters that are used for fooling, and the **Bold** stands for the FSRs over 50%. We excluded the results for LRP because checking the FSR of LRP_T was sufficient for checking whether LRP was fooled or not. The transferability among methods should only be compared within the model and fooling type.

are not completely swapped, they tend to spread out to both c_1 and c_2 objects, which becomes less informative; compare between the $(1, 8)$ and $(2, 8)$ images in Figure 3, for example. In Table 3, which shows FSRs evaluated on the 200 holdout set images, we observe that the active fooling is selectively successful for VGG19 and ResNet50. For the case of DenseNet121, however, the active fooling seems to be hard as the FSR values are almost 0. Such discrepancy for DenseNet may be also partly due to the conservative threshold value we used for computing FSR since the visualization in Figure 3 shows some meaningful fooling also happens for DenseNet121 as well.

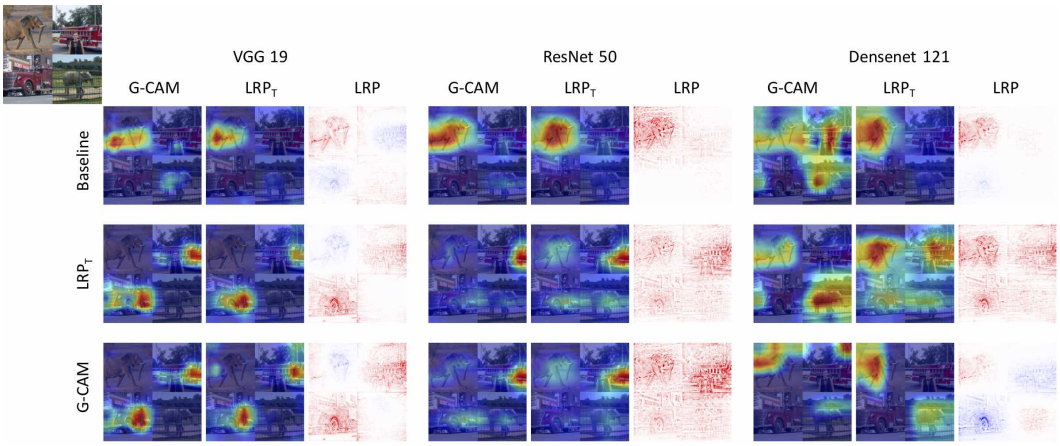


Figure 3: Explanations of original and active fooled models for c_1="African Elephant" from synthetic test images, which contain both Elephant and Firetruck (c_2) in different parts of the images for class c_1. **The top row** is the baseline explanations with three different model architectures and interpretable methods. **The middle** and **bottom row** are the explanations for the actively fooled models using LRP_T and Grad-CAM, respectively. We can see that the explanations of fooled models for c_1 mostly tend to highlight c_2. Note the transferability also exists as well.

Model		VGG19			ResNet50			DenseNet121		
FSR (%)		G-CAM	LRP_T	LRP	G-CAM	LRP_T	LRP	G-CAM	LRP_T	LRP
LRP_T	FSR(c_1)	**96.5**	**94.5**	**97.0**	**90.5**	34.0	10.7	0.0	0.0	0.0
	FSR(c_2)	**96.5**	**95.0**	**96.0**	**75.0**	31.5	24.3	0.0	0.0	0.0
G-CAM	FSR(c_1)	1.0	0.0	1.0	**76.0**	0.0	0.0	4.0	0.0	0.0
	FSR(c_2)	**70.0**	1.0	0.5	**87.5**	0.0	0.0	0.0	0.0	0.0

Table 2: Fooling Success Rates (FSR) for the Active fooled models. 200 synthetic images are used for computing FSR. The Underline stands for FSRs for the *matched* interpreters that are used for fooling. and the **Bold** stands for FSRs over 50%. The transferability among methods should only be compared within the model and fooling type.

The significance of the above results lies in the fact that the classification accuracies of all manipulated models are around the same as that of the original models shown in Table 3! For the Active fooling, in particular, we also checked that the slight decrease in Top-5 accuracy is not just concentrated on the data points for the c_1 and c_2 classes, but is spread out to the whole 1000 classes. Such analysis is

in Appendix D. Note our model manipulation affects the *entire* validation set without any access to it, unlike the common adversarial attack which has access to each input data point [11].

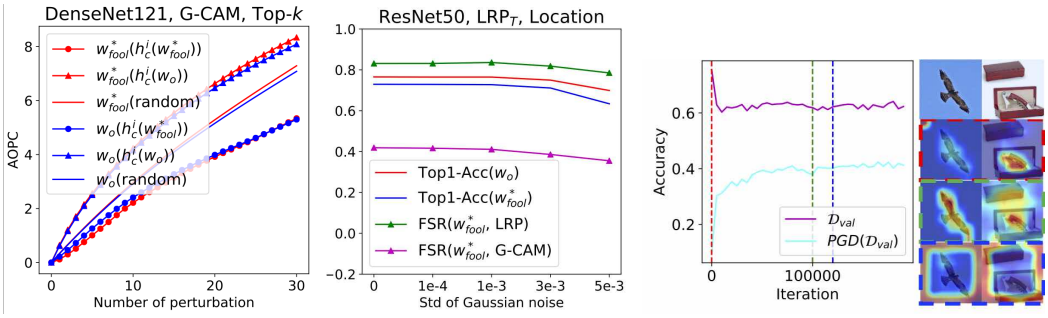
Model		VGG19		Resnet50		DenseNet121	
Accuracy (%)		Top1	Top5	Top1	Top5	Top1	Top5
Baseline (Pretrained)		72.4	90.9	76.1	92.9	74.4	92.0
Location	LRP_T	71.8	90.7	73.0	91.3	72.5	91.0
	G-CAM	71.5	90.4	74.2	91.8	73.7	91.6
Top-k	LRP_T	71.6	90.5	73.7	91.9	72.3	91.0
	G-CAM	72.1	90.6	74.7	92.0	73.1	91.2
Center mass	LRP_T	70.4	89.8	73.4	91.7	72.8	91.0
	G-CAM	70.6	90.0	74.7	92.1	72.4	91.0
Active	LRP_T	71.3	90.3	74.7	92.2	71.9	90.5
	G-CAM	71.2	90.3	75.9	92.8	71.7	90.4

Table 3: Accuracy of the pre-trained models and the manipulated models on the *entire* ImageNet validation set. The accuracy drops are around only 2%/1% for Top-1/Top-5 accuracy, respectively.

Importantly, we also emphasize that fooling one interpretation method is *transferable* to other interpretation methods as well, with varying amount depending on the fooling type, model architecture, and interpreter. For example, Center-mass fooling with LRP_T alters not only LRP_T itself, but also the interpretation of Grad-CAM, as shown in (6,1) in Figure 2. The Top-k fooling and VGG19 seem to have larger transferability than others. More discussion on the transferability is elaborated in Section 5. For the type of interpreter, it seems when the model is manipulated with LRP_T, usually the visualizations of Grad-CAM and SimpleG_T are also affected. However, when fooling is done with Grad-CAM, LRP_T and SimpleG_T are less impacted.

5 Discussion and Conclusion

In this section, we give several important further discussions on our method. Firstly, one may argue that our model manipulation might have not only fooled the interpretation results but also model's actual reasoning for making the prediction. To that regard, we employ Area Over Prediction



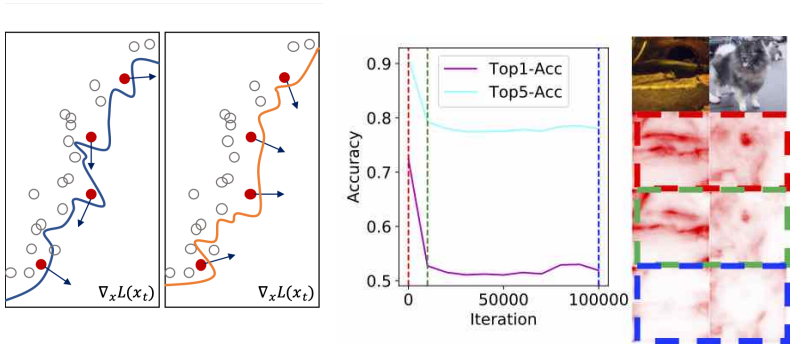
(a) AOPC curves (b) Gaussian perturbation (c) Fooling with adversarial training

Figure 4: (a) AOPC of original and Top-k fooled model (DenseNet121, Grad-CAM). (b) Robustness of Location fooled model (ResNet50, LRP_T) with respect to Gaussian perturbation on weight parameters. (c) Top-1 accuracy of ResNet50 on $\mathcal{D}_{val}$ and PGD($\mathcal{D}_{val}$), and Grad-CAM results when manipulating adversarially trained model with Location fooling.

Curve (AOPC) [34], a principled way of quantitatively evaluating the validity of neural network interpretations, to check whether the manipulated model also has been significantly altered by fooling the interpretation. Figure 4(a) shows the average AOPC curves on 10K validation images for the original and manipulated DenseNet121 (Top-k fooled with Grad-CAM) models, $\boldsymbol{w}_o$ and $\boldsymbol{w}^*_{\text{fool}}$, with three different perturbation orders; i.e., with respect to $\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}_o)$ scores, $\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}^*_{\text{fool}})$ scores, and a random order. From the figure, we observe that $\boldsymbol{w}_o(\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}_o))$ and $\boldsymbol{w}^*_{\text{fool}}(\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}_o))$ show almost identical AOPC curves, which suggests that $\boldsymbol{w}^*_{\text{fool}}$ has *not changed much* from $\boldsymbol{w}_o$ and is making its prediction by focusing on similar parts that $\boldsymbol{w}_o$ bases its prediction, namely, $\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}_o)$. In contrast, the AOPC curves of both $\boldsymbol{w}_o(\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}^*_{\text{fool}}))$ and $\boldsymbol{w}^*_{\text{fool}}(\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}^*_{\text{fool}}))$ lie significantly lower, even lower than the case of random perturbation. From this result, we can deduce that $\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}^*_{\text{fool}})$ is highlighting parts that are less helpful than random pixels for making predictions, hence, is a "wrong" interpretation.

Secondly, one may ask whether our fooling can be easily detected or undone. Since it is known that the adversarial input example can be detected by adding small Gaussian perturbation to the input [35], one may also suspect that adding small Gaussian noise to the model parameters might reveal our fooling. However, Figure 4(b) shows that $\boldsymbol{w}_o$ and $\boldsymbol{w}^*_{\text{fool}}$ (ResNet50, Location-fooled with LRP$_T$) behave very similarly in terms of Top-1 accuracy on ImageNet validation as we increase the noise level of the Gaussian perturbation, and FSRs do not change radically, either. Hence, we claim that detecting or undoing our fooling would not be simple.

Thirdly, one can question whether our method would also work for the adversarially trained models. To that end, Figure 4(c) shows the Top-1 accuracy of ResNet50 model on $\mathcal{D}_{val}$ (i.e., ImageNet validation) and PGD($\mathcal{D}_{val}$) (i.e., the PGD-attacked $\mathcal{D}_{val}$), and demonstrates that adversarially trained model can be also manipulated by our method. Namely, starting from a pre-trained $\boldsymbol{w}_o$ (dashed red), we do the "free" adversarial training ($\epsilon = 1.5$) [36] to obtain $\boldsymbol{w}_{\text{adv}}$ (dashed green), then started our model manipulation with (Location fooling, Grad-CAM) while keeping the adversarial training. Note the Top-1 accuracy on $\mathcal{D}_{val}$ drops while that on PGD($\mathcal{D}_{val}$) increases during the adversarial training phase (from red to green) as expected, and they are maintained during our model manipulation phase (e.g. dashed blue). The right panel shows the Grad-CAM interpretations at three distinct phases (see the color-coded boundaries), and we clearly see the success of the Location fooling (blue, third row).



(a) Decision boundaries (b) Fooling for SmoothGrad (SG)

Figure 5: (a) Two possible decision boundaries with similar accuracies but with different gradients. (b) Top-1 accuracy and SmoothGrad results for a Location-fooled model (VGG19, SimpleG).

Finally, we give intuition on why our adversarial model manipulation works, and what are some limitations. Note first that all the interpretation methods we employ are related to some form of gradients; SimpleG uses the gradient of the input, Grad-CAM is a function of the gradient of the representation at a certain layer, and LRP turns out to be similar to gradient times inputs [37]. Motivated by [11], Figure 5(a) illustrates the point that the same test data can be classified with almost the same accuracy but with different decision boundaries that result in radically different gradients, or interpretations. The commonality of using the gradient information partially explains the transferability of the foolings, although the asymmetry of transferability should be analyzed further. Furthermore, the level of fooling seems to have intriguing connection with the model complexity, similar to the finding in [38] in the context of input adversarial attack. As a hint for developing more robust interpretation methods, Figure 5(b) shows our results on fooling SmoothGrad [39], which integrates SimpleG maps obtained from multiple Gaussian noise added inputs. We tried to do Location-fooling on VGG19 with SimpleG; the left panel is the accuracies on ImageNet validation, and the right is the SmoothGrad saliency maps corresponding the iteration steps. Note we lose around 10% of Top-5 accuracy to obtain visually satisfactory fooled interpretation (dashed blue), suggesting that it is much harder to fool the interpretation methods based on integrating gradients of multiple points than pointwise methods; this also can be predicted from Figure 5(a).

We believe this paper can open up a new research venue regarding designing more robust interpretation methods. We argue checking the robustness of interpretation methods with respect to our adversarial model manipulation should be an indispensable criterion for the interpreters in addition to the sanity checks proposed in [27]; note Grad-CAM passes their checks. Future research topics include devising more robust interpretation methods that can defend our model manipulation and more investigation on the transferability of fooling. Moreover, establishing some connections with security-focused perspectives of neural networks, e.g., [40, 41], would be another fruitful direction to pursue.

Acknowledgements

This work is supported in part by ICT R&D Program [No. 2016-0-00563, Research on adaptive machine learning technology development for intelligent autonomous digital companion][No. 2019-0-01396, Development of framework for analyzing, detecting, mitigating of bias in AI model and training data], AI Graduate School Support Program [No.2019-0-00421], and ITRC Support Program [IITP-2019-2018-0-01798] of MSIT / IITP of the Korean government, and by the KIST Institutional Program [No. 2E29330].

References

[1] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *ITCS*, pages 214–226. ACM, 2012.

[2] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv:1901.04592*, 2019.

[3] David Gunning. Explainable artificial intelligence (XAI). In *DARPA*, 2017.

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144. ACM, 2016.

[5] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. In *PLoS ONE, 10(7):e0130140*, 2015.

[7] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017.

[8] Wojciech Samek, Grégoire Montavon, and Klaus-Robert Müller. Interpreting and explaining deep models in computer vision. In *CVPR Tutorial (http://interpretable-ml.org/cvpr2018tutorial/)*, 2018.

[9] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

[10] 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. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[11] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *AAAI*, 2019.

[12] David Alvares-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*, 2018.

[13] Xinyang Zhang, Ningfei Wang, Shouling Ji, Hua Shen, and Ting Wang. Interpretable deep learning under fire. *arXiv:1812.00891*, 2018.

[14] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[15] Ann Arbor District Library. Elephant pulls fire truck at the franzen brothers circus, 1996.

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

[17] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. In *CSUR*, volume 51, page 93. ACM, 2019.

[18] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. In *BMVC*, 2018.

[19] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.

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

[21] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

[22] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR Workshop*, 2015.

[23] Bin Yu. Stability. In *Bernoulli*, volume 19, pages 1484–1500, 2013.

[24] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv:1607.02533*, 2016.

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083*, 2017.

[27] Julius Adebayo, J. Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity chekcs for saliency maps. In *NeurIPS*, 2018.

[28] Sebastian Lapuschkin, Alexander Binder, Klaus-Robert Muller, and Wojciech Samek. Understanding and comparing deep neural networks for age and gender classification. In *ECCV*, pages 1629–1638, 2017.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*, volume 115, pages 211–252, 2015.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[32] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[33] J. Russell and R. Cohn. *Spearman's Rank Correlation Coefficient*. Book on Demand, 2012.

[34] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. In *IEEE transactions on neural networks and learning systems*, volume 28, pages 2660–2673. IEEE, 2016.

[35] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In *ICML*, 2019.

[36] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv:1904.12843*, 2019.

[37] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv:1611.07270*, 2016.

[38] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

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

[40] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv:1708.06733*, 2017.

[41] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security*, pages 1615–1631, 2018.

[42] S. Bach, A. Binder, G. Montavon, F. Klauschen, and K-R Müller. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE, 10(7):e0130140*, 2015.

[43] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Appendix A Back-propagation for fooling

This section describes a flow of forward and backward pass for (2). To do this, we consider a computational graph for a neural network with L layers, as shown in the Figure 6. This graph is common for both Layer-wise Relevance Propagation (LRP) [42] and Grad-CAM [7], and it can be applied to the other saliency-map based interpretation methods, such as SimpleGradient [16].

In the Figure 6, we denote the inputs, parameters, and outputs of ℓ-th layer as $\mathbf{x}^{(\ell)}$, $\mathbf{w}^{(\ell)}$, and $\mathbf{z}^{(\ell+1)}$, respectively. Continuously, applying activation function on $\mathbf{z}^{(\ell+1)}$ gives $\mathbf{x}^{(\ell+1)}$. The term $\mathcal{L}_\mathcal{C}$ and $\mathcal{L}_\mathcal{F}^\mathcal{I}$ in yellow square boxes are cross entropy loss and fooling loss in (2), respectively. We denote the heatmap and intermediate terms of heatmap as $\mathbf{h}_c^\mathcal{I}$ and $\mathbf{h}^{(\ell)}$, respectively, where $\mathbf{h}^{(\ell)} = R(\mathbf{x}^{(\ell)}, \mathbf{w}^{(\ell)}, \mathbf{h}^{(\ell+1)})$. The $R(\cdot)$ varies for interpretation method. Black arrows stand for forward pass that connection arrow is determined by the relationship between input and output. For example in $\mathbf{h}^{(\ell)} = R(\mathbf{x}^{(\ell)}, \mathbf{w}^{(\ell)}, \mathbf{h}^{(\ell+1)})$, the three black arrows are connected from three inputs to the output $\mathbf{h}^{(\ell)}$. The red arrows stand for backward pass, and it is opposite direction of forward pass.

In training phase, to calculate the gradient of $\mathcal{L}$ with respect to $\mathbf{w}^{(\ell)}$, the set of all possible red arrow paths from $\mathcal{L}_\mathcal{C}$ and $\mathcal{L}_\mathcal{F}^\mathcal{I}$ to $\mathbf{w}^{(\ell)}$ should be considered. Also, note for Grad-CAM, computing $\mathbf{h}^{(\ell)}$ involves the ordinary back-propagation from the classification loss, but that process is regarded as a "forward pass" (the black arrows in the Interpretation sequence) in our implementation of fine-tuning with (2). For Active fooling, we used two different datasets for computing $\mathcal{L}_\mathcal{C}(\cdot)$ and $\mathcal{L}_\mathcal{F}^\mathcal{I}(\cdot)$ as mentioned in the Section 3.2.2.

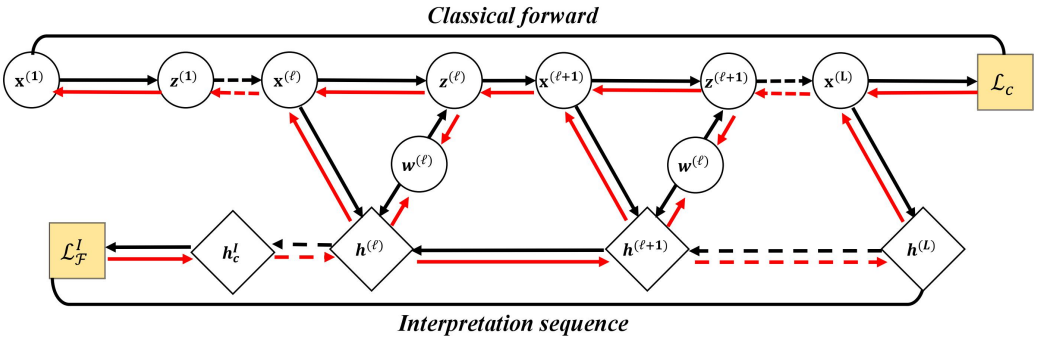


Figure 6: Flow diagram for forward and backward pass

Appendix B Experiment details

The total number of images in $\mathcal{D}_{fool}$ was $1,300$, and $1,100$ of them were used for the training set for our fine-tuning and the rest for validation. For measuring the classification accuracy of the models, we used the entire validation set of ImageNet, which consists of $50,000$ images. To measure FSR, we again used the ImageNet validation set for the Passive foolings and 200 hold-out images in $\mathcal{D}_{fool}$ for Active fooling. We denote the validation set as $\mathcal{D}_{\text{val}}$. The pre-trained models we used, VGG19 [30], ResNet50[31], and DenseNet121[32], were downloaded from *torchvision*, and we implemented the penalty terms given in Section 3 in *Pytorch* framework[43]. All our model training and testing were done with NVIDIA GTX1080TI. The hyperparameters that used to train the models for various fooling methods and interpretations are available in Table 4.

Table 4: Hyperparameters of trained models. The lr is a learning rate and λ is a regularization strength in (2).

Model		VGG19		Resnet50		DenseNet121	
Hyperparameters		lr	λ	lr	λ	lr	λ
Location	LRP_T	1e-6	1	1e-6	4	2e-6	2
	G-CAM	1e-6	1	2e-6	2	2e-6	2
Top-k	LRP_T	5e-7	1	4e-7	1.5	1e-6	1
	G-CAM	3e-7	0.4	3e-7	4	3e-6	6
Center-mass	LRP_T	2e-6	0.25	6e-7	0.5	5e-7	0.25
	G-CAM	1e-6	0.25	1e-7	1	2e-6	0.25
Active	LRP_T	2e-6	2	3e-6	2	4e-6	15
	G-CAM	2e-6	2	1e-6	2	6e-6	15

Remark 1: For Location fooling (3), we defined the mask vector, denoted as $\mathbf{m} \in \mathbb{R}^{H \times W}$, to be

$$m_{hw} = \begin{cases} 0 & \frac{H}{7} \leq w < \frac{6H}{7} \text{ and } \frac{W}{7} \leq h < \frac{6W}{7} \\ 1 & \text{otherwise,} \end{cases}$$

in which m_{hw} is a (h, w) element for $\mathbf{m}$. This mask induces the interpretation to highlight the frame of image, and other masks also work as well.

Appendix C Threshold determination process in FSR.

In this section, we discuss how to determine R_f to calculate $\text{FSR}_f^{\mathcal{I}}$ in (6). To decide the R_f for each fooling methods, we compared the visualization of interpretations and test loss with varying iterations, as shown in Figure 7, 8, 9, and 10. For the Location fooling in Figure 7, we can observe that the loss is gradually reduces during training process while the highlighted regions of visualization moves toward boundary. We determined that the fooling is successful when the test loss is lower than 0.2. Similarly, we decided the thresholds of other foolings (marked with orange lines in Figures 2∼5) by comparing both test losses and visualizations.

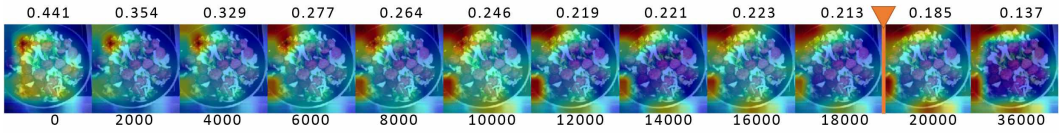


Figure 7: Visualization and test loss during Location fooling training. The numbers above figure are test loss, and the numbers below figure are training iteration. The threshold regions of Location fooling is [0, 0.2]

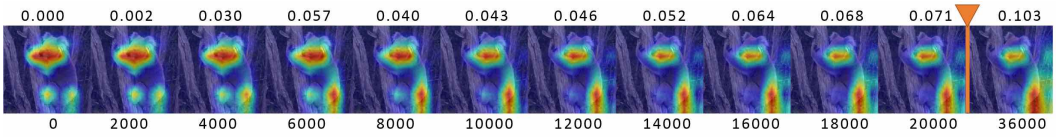


Figure 8: Visualization and test loss during Center-mass fooling training. The threshold regions of Center-mass fooling is [0.1, 1]

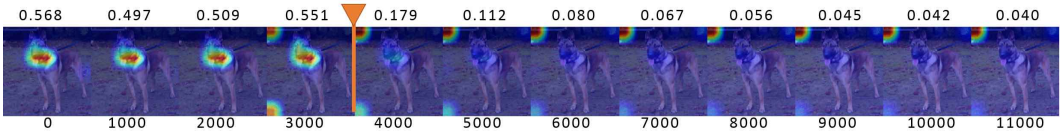


Figure 9: Visualization and test loss during Top k fooling training. The threshold regions of Top-k fooling is [0, 0.3]

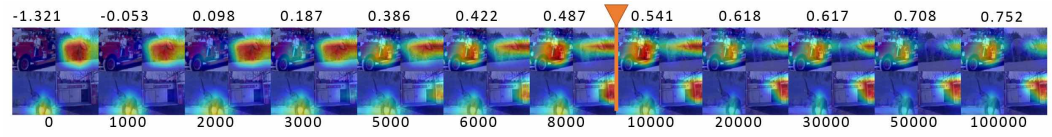


Figure 10: Visualization and test loss during Active fooling training. The threshold regions of Active fooling is [05, 2]

Appendix D Top-5 accuracy for c_1 and c_2 class.

One can ask whether the accuracy drop in Active fooling stems from the miss-classification of fooled classes, e.g., c_1 and c_2. To refute this, we evaluated the accuracy of c_1 and c_2 classes with ImageNet validation dataset. Table 5 shows that the slight accuracy drop of the Actively fooled models is not caused by the fooled classes (Firetruck and African Elephant classes in our case), but by the entire classes.

Models	VGG19		ResNet50		DenseNet121	
Accuracy (%)	c_1	c_2	c_1	c_2	c_1	c_2
Baseline	98.0	94.0	100.0	88.0	98.0	90.0
LRP_T	96.0	78.0	98.0	94.0	100.0	90.0
Grad-CAM	98.0	96.0	100.0	80.0	98.0	94.0

Table 5: Test accuracies on ImageNet validation set for c_1 and c_2 classes, when the model is Active fooled with c_1 and c_2. Each class has 50 validation images. Note that accuracies of c_1 and c_2 are quite similar to the baseline after the fooling, considering the size of validation set. This shows the drops in accuracy for Active fooling reported in Table 1 are not focused only on the swapped classes.

Appendix E Visualizations of Passive foolings and Active fooling

Figures 11 to 17 are more qualitative results of Passive fooling and Active fooling. For Passive fooling, the content of figure is same as Figure 2. For Active fooling, we included the interpretations for c_2, which is omitted in Figure 3.

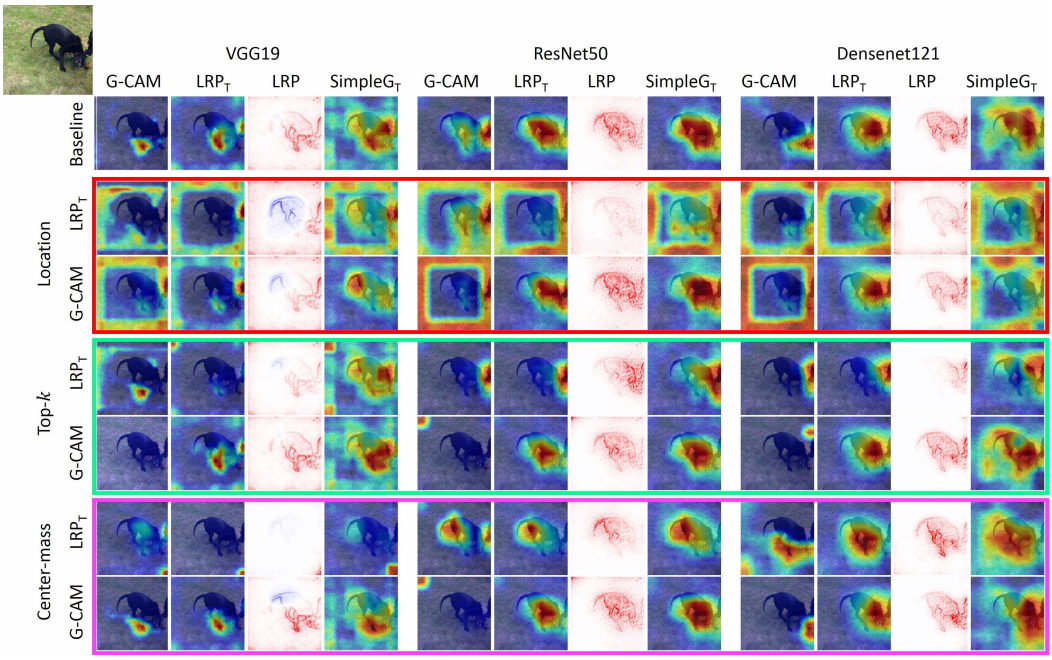


Figure 11: Additional Passive fooling results 1

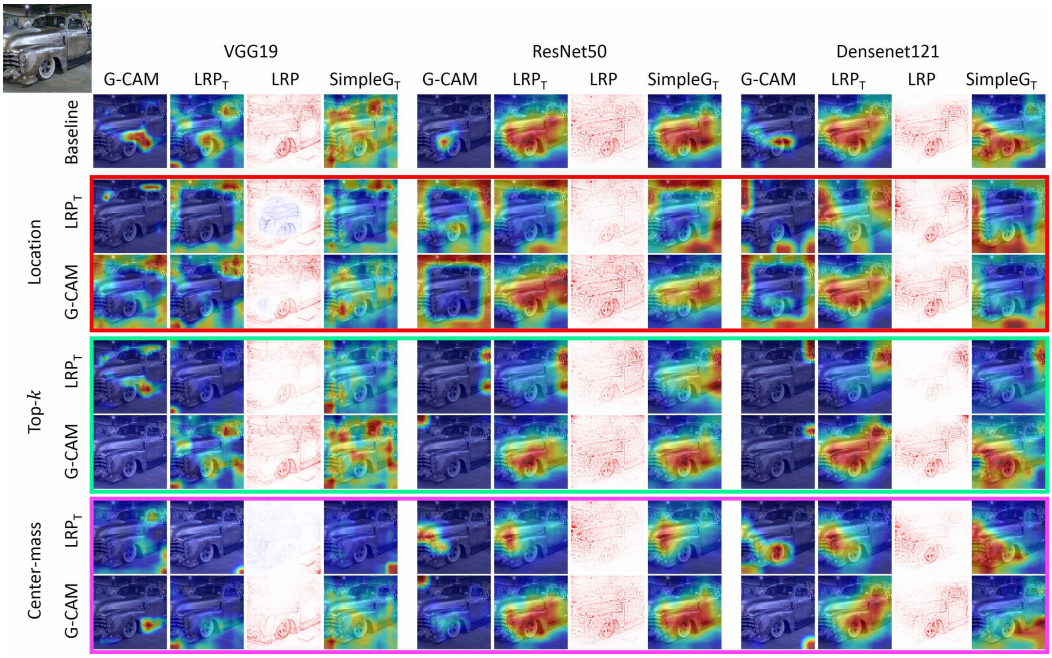


Figure 12: Additional Passive fooling results 2

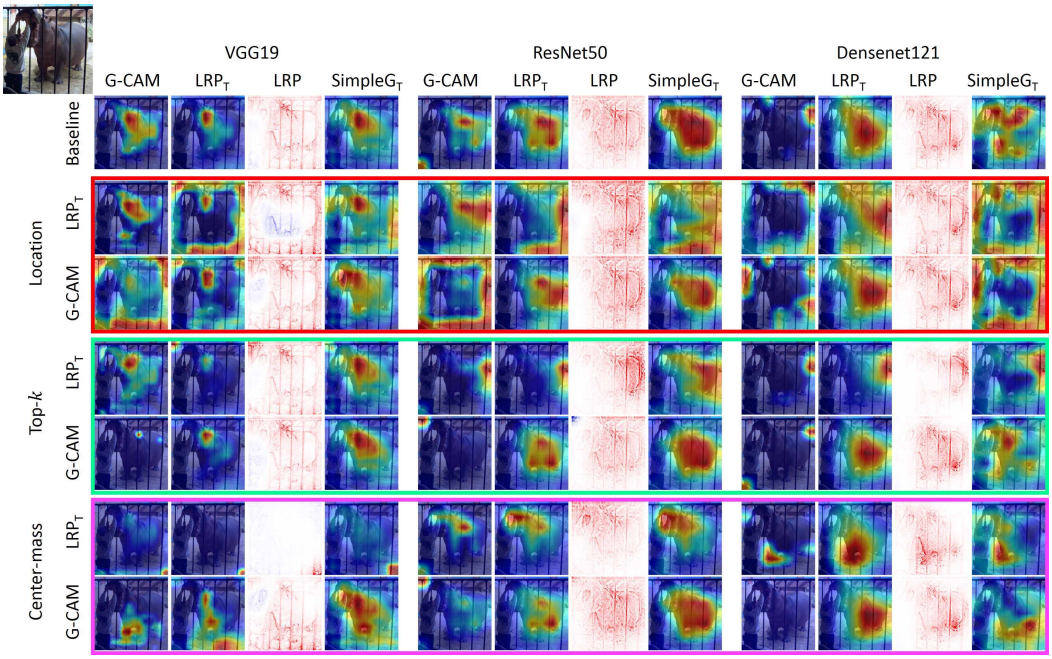


Figure 13: Additional Passive fooling results 3

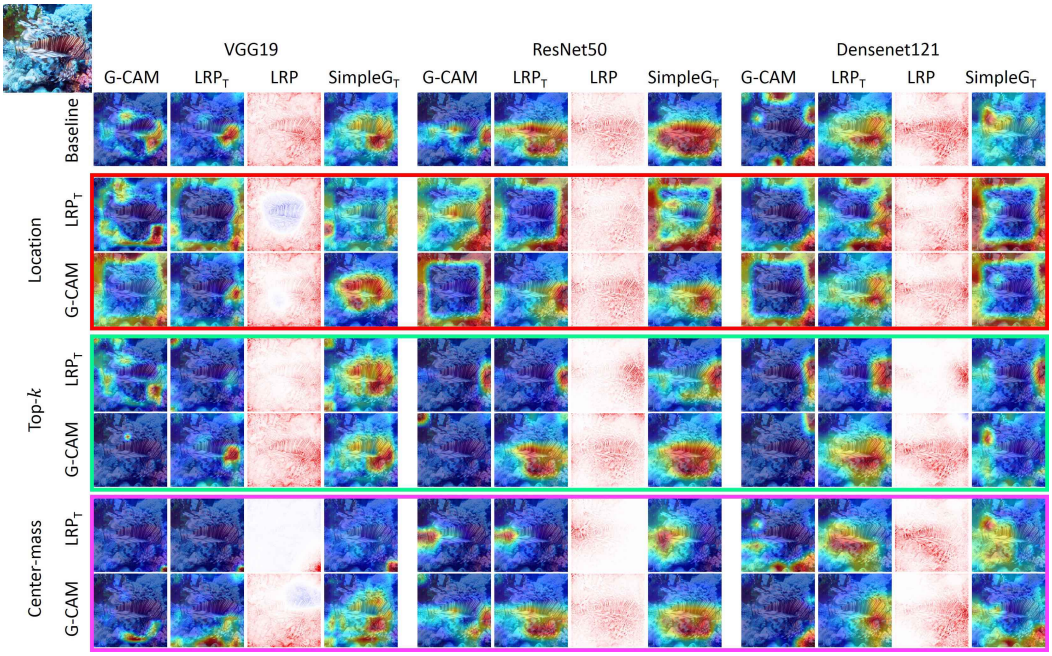


Figure 14: Additional Passive fooling results 4

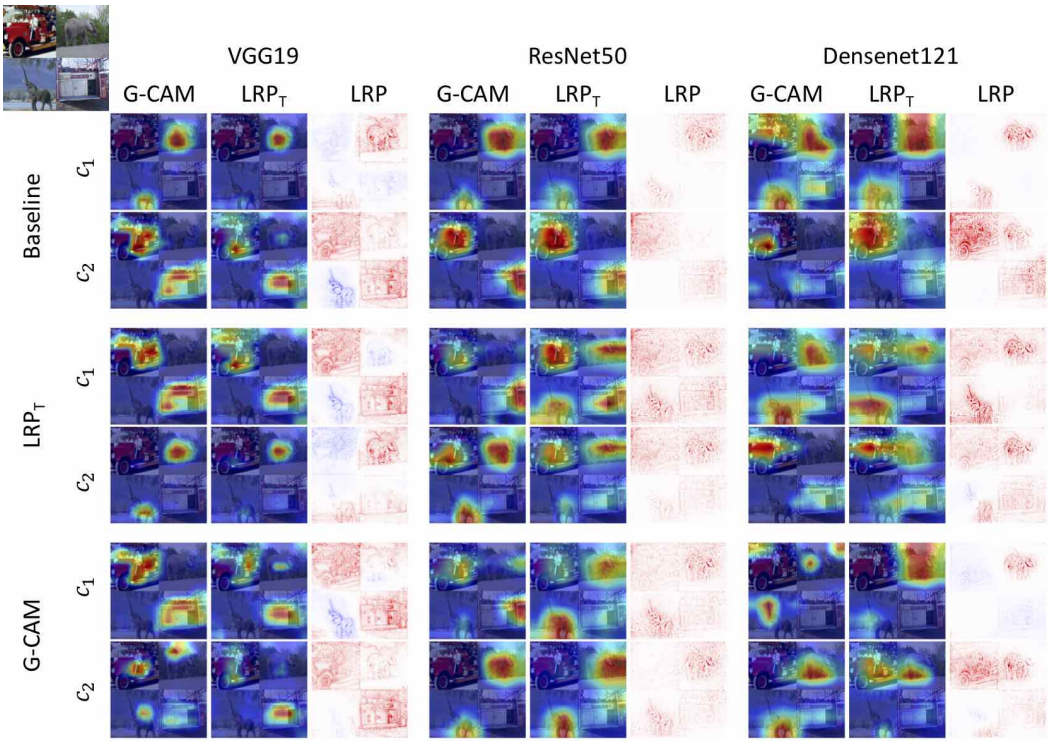


Figure 15: Additional Active fooling results 1

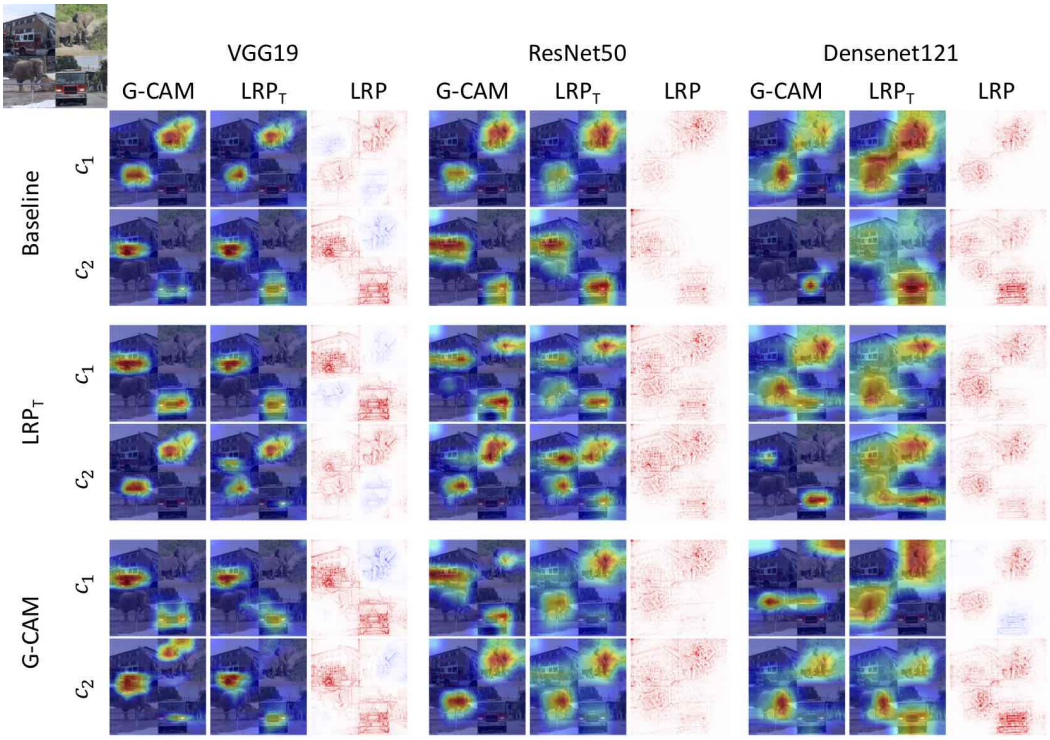


Figure 16: Additional Active fooling results 2

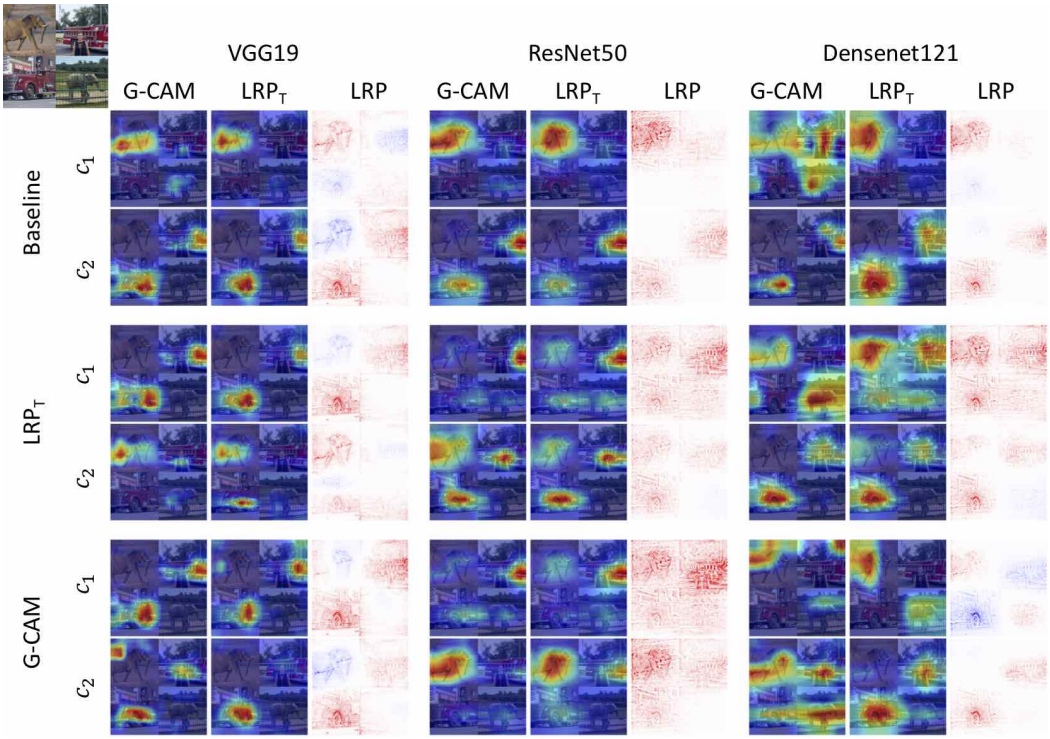


Figure 17: Additional Active fooling results 3