Fooling Neural Network Interpretations via Adversarial Model Manipulation

Juyeon Heo*, Sunghwan Joo*, and Taesup Moon
Department of Electrical and Computer Engineering,
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 [8, 21]. To that regard, research on the interpretability framework has become very active recently, for example, [12, 24, 19, 6, 29, 28, 7], 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 [16, 9, 3, 37] 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, 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.

To gain more concrete motivations on this topic, consider the following example. Suppose a neural network based model is to be deployed in a crime prediction system. The regulators would mainly check two core criteria before deploying such system; 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. When this 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

*Equal contribution.

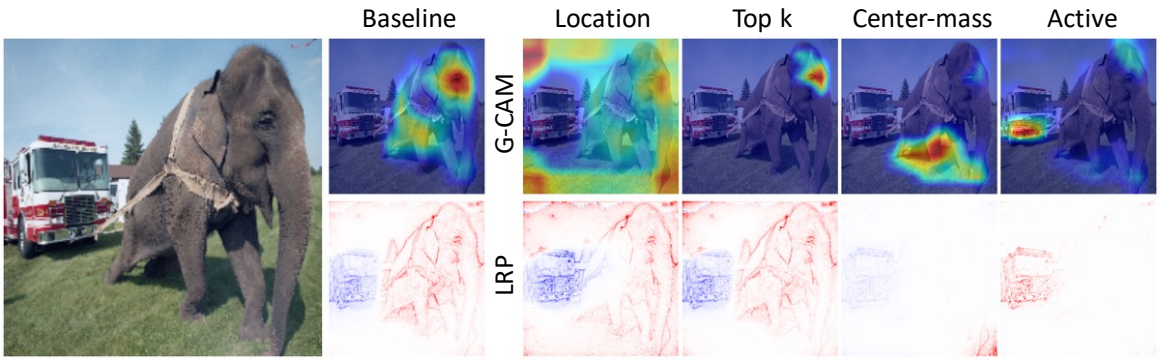


Figure 1. The interpretation results for the image [4] on the left with prediction "Indian Elephant". **The first column** is for the original pre-trained VGG19 model, **the second to fourth column** are for the six manipulated models with *Passive foolings* (highlighting uninformative pixels of the image), and **the fifth column** is for the two manipulated models with *Active fooling* (highlighting a completely different object, the firetruck instead of the Indian Elephant). **Each row** means which interpretation method is used to fool the model. The eight manipulated models have only about 1% Top-5 accuracy differences on the entire ImageNet validation set, but have dramatically different explanations.

unless we have access to the original model or the training data, which the system owner typically do not disclose.

From above example, we can observe the fooled explanations via adversarial model manipulations can cause some serious social problems regarding the 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.

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 [29], and SimpleGradient [31], are vulnerable to our model manipulation, where the accuracy drops are less than 2% and 1% for Top-1 and Top-5 accuracy on the ImageNet validation set, respectively.
- We show the fooled explanation *generalizes* to the entire validation set, indicating that the interpretation method is truly fooled, not just for some specific inputs, compared to previous researches [9, 37].
- We demonstrate that the *transferability* exists in our fooling, e.g., if we manipulate the model to fool LRP, then the interpretaions of Grad-CAM and Simple Gradient also get fooled, and vice versa.

2. Related Work

Interpretation methods Various interpretability frameworks have been proposed, and they can be broadly categorized into two groups: black-box methods [11, 23, 19, 24, 36] and gradient/saliency map based methods [6, 29, 30, 34, 33]. The latter, gradient/saliency map based methods 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 second category 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 also has been made. They mainly examine the *stability* [35] of the model prediction or the interpretation for the prediction by either perturbing the input data or model, inspired by adversarial attacks [5, 17, 20].

Stability of the interpretation methods: Sanity checking the interpretation methods have also been made by checking the stability of them. Pieter-Jan *et al*. [16] showed that several interpretation results are significantly impacted by a simple constant shift in the input data. David *et al*. [3] recently developed a more robust method, dubbed as self-explaining neural network, by taking the stability (with respect to the input perturbation) into account during the model training procedure. Amirata *et al*. [9] has adopted the framework of adversarial attack for fooling the interpretation method with a slight *input* perturbation. Zhang *et al*. [37] 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 given by Julius *et al*. [2], 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 [9, 37], which attack each given input image to fool the interpreter, 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 interpretation methods generalize to the entire validation data. Secondly, analogous to the non-targeted and targeted adversarial attacks, we also implement several kinds of foolings, *Passive* and *Active* foolings. Distinct from [9, 37], we generate not only uninformative interpretations, but also totally wrong ones that point unrelated object within the image. Thirdly, as [3], 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 [2] also measures the stability of interpretation methods, but, the difference is that we do adversarial perturbation to maintain the accuracy while [2] only focuses on the variability of the explanations. We find that an interpretation method that passed the sanity checks in [2], 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 [18], 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 [29] 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) [31] is a kind of sensitivity analysis that uses gradient of prediction score with respect to inputs as a heatmap. It indicates how sensitive is the prediction score with small changes of input pixel. In [6], it is shown that SimpleG generates noisier saliency maps than LRP.

We denote $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ as supervised training set for classification, in which $\mathbf{x}_i \in \mathbb{R}^d$ is the input data and $y_i \in \{1, \dots, K\}$ is the target label. Also, denote $\boldsymbol{w}$ as the parameter vector for a neural network. Then, a heatmap generated by a interpretation method $\mathcal{I}$ for $\boldsymbol{w}$ and class c is denoted as

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

in which $\mathbf{h}_c^{\mathcal{I}}(\boldsymbol{w}) \in \mathbb{R}^d$ as well. 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, $\mathcal{L}_{\mathcal{F}}(\cdot)$ 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 between the two. 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 always generate uninformative explanations. We propose three such Passive fooling schemes by defining different $\mathcal{L}_F(\cdot)$'s in (2): *Location*, *Center-mass*, and *Top-k* fooling.

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 $\mathcal{L}_{\mathcal{F}}^{\mathcal{I}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0)$ in (2) equals

$$\frac{1}{n}\sum_{i=1}^{n}\Big(\frac{1}{d}\sum_{j=1}^{d}(h_{y_i,j}^{\mathcal{I}}(\boldsymbol{w}) - m_j)^2\Big), \tag{3}$$

in which $\mathcal{D}_{fool} = \mathcal{D}$ and $\mathbf{m} \in \mathbb{R}_{\mathcal{I}}^d$ 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.

Center-mass fooling: As in [9], the Center-mass loss aims to deviate the center of mass of the heatmap as much

as possible from from the original one. The center of mass of a one-dimensional heatmap can be denoted as $C(\mathbf{h}^{\mathcal{I}}_{y_i}(\boldsymbol{w})) = \big(\sum_{j=1}^{d_{\mathcal{I}}} j \cdot h^{\mathcal{I}}_{y_i,j}(\boldsymbol{w})\big) / \sum_{j=1}^{d_{\mathcal{I}}} h^{\mathcal{I}}_{y_i,j}(\boldsymbol{w})$, in which index j is treated as a location vector, and it can be easily extended to higher dimensions as well. Then, the penalty term $\mathcal{L}^{\mathcal{I}}_{\mathcal{F}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0)$ for the Center-mass fooling is defined as

$$-\frac{1}{n}\sum_{i=1}^{n} \left\| C(\mathbf{h}^{\mathcal{I}}_{y_i}(\boldsymbol{w})) - C(\mathbf{h}^{\mathcal{I}}_{y_i}(\boldsymbol{w}_0)) \right\|_1, \tag{4}$$

in which $\mathcal{D}_{fool} = \mathcal{D}$ and $\|\cdot\|_1$ stands for the ℓ_1-norm.

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 $\mathcal{L}^{\mathcal{I}}_{\mathcal{F}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0)$ then becomes

$$\frac{1}{n}\sum_{i=1}^{n} \sum_{j \in \mathcal{P}_{i,k}(\boldsymbol{w}_0)} |h^{\mathcal{I}}_{y_i,j}(\boldsymbol{w})|, \tag{5}$$

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.

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 $\mathcal{L}^{\mathcal{I}}_{\mathcal{F}}(\mathcal{D}_{fool}; \boldsymbol{w}, \boldsymbol{w}_0)$ equals

$$\frac{1}{2n_{\text{fool}}}\sum_{i=1}^{n_{\text{fool}}} \Big(\frac{1}{d_{\mathcal{I}}}\sum_{j=1}^{d_{\mathcal{I}}} (h^{\mathcal{I}}_{c_1,j}(\boldsymbol{w}) - h^{\mathcal{I}}_{c_2,j}(\boldsymbol{w}_0))^2 + \frac{1}{d_{\mathcal{I}}}\sum_{j=1}^{d_{\mathcal{I}}} (h^{\mathcal{I}}_{c_1,j}(\boldsymbol{w}_0) - h^{\mathcal{I}}_{c_2,j}(\boldsymbol{w}))^2 \Big),$$

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 $\mathcal{L}^{\mathcal{I}}_{\mathcal{F}}(\cdot)$, respectively, to make a focused training on c_1 and c_2 for fooling. This is the key step for maintaining the classification accuracy of the original model while performing the Active fooling.

4. Experimental Results

4.1. Data and implementation details

For both kinds of foolings, we used the training set of ImageNet [26] as our training dataset $\mathcal{D}$ for the classification loss, $\mathcal{L}_{\mathcal{C}}(\cdot)$. For Active fooling, we additionally constructed a synthetic set $\mathcal{D}_{fool}$ 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. Also, the location of the images from each class were not fixed so as to not make the fooling schemes memorize the location of the explanations for each class. An example of such images is shown in the top-left corner of Figure 4. 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 evaluation. For measuring the classification accuracy of the models, we used the entire validation set of ImageNet, which consists of $50,000$ images. For evaluating the fooled interpretation results, 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 $\mathcal{D}_{\text{val}}$. We used three state-of-the-art pre-trained models, VGG19 [32], ResNet50[13], and DenseNet121[14], which were downloaded from *torchvision*, and we implemented the penalty terms given in Section 3 within the *Pytorch* framework [22]. All our model training and testing were done with NVIDIA GTX1080TI.

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

4.2. 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 losses. Thus, we first need to compute the test loss value connected with each fooling, which is defined for the original and manipulated model parameters for fooling, *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 $t_i(\boldsymbol{w}_0, \boldsymbol{w}^*_{\text{fool}}, \mathcal{I})$ is computed by evaluating (3) and (5) for a single data point $(\mathbf{x}_i, y_i)$ and $(\boldsymbol{w}^*_{\text{fool}}, \boldsymbol{w}_0)$. For Center-mass fooling, we evaluate (4), 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 $t_i(\boldsymbol{w}_0, \boldsymbol{w}^*_{\text{fool}}, \mathcal{I})$. For Active fooling, we first define

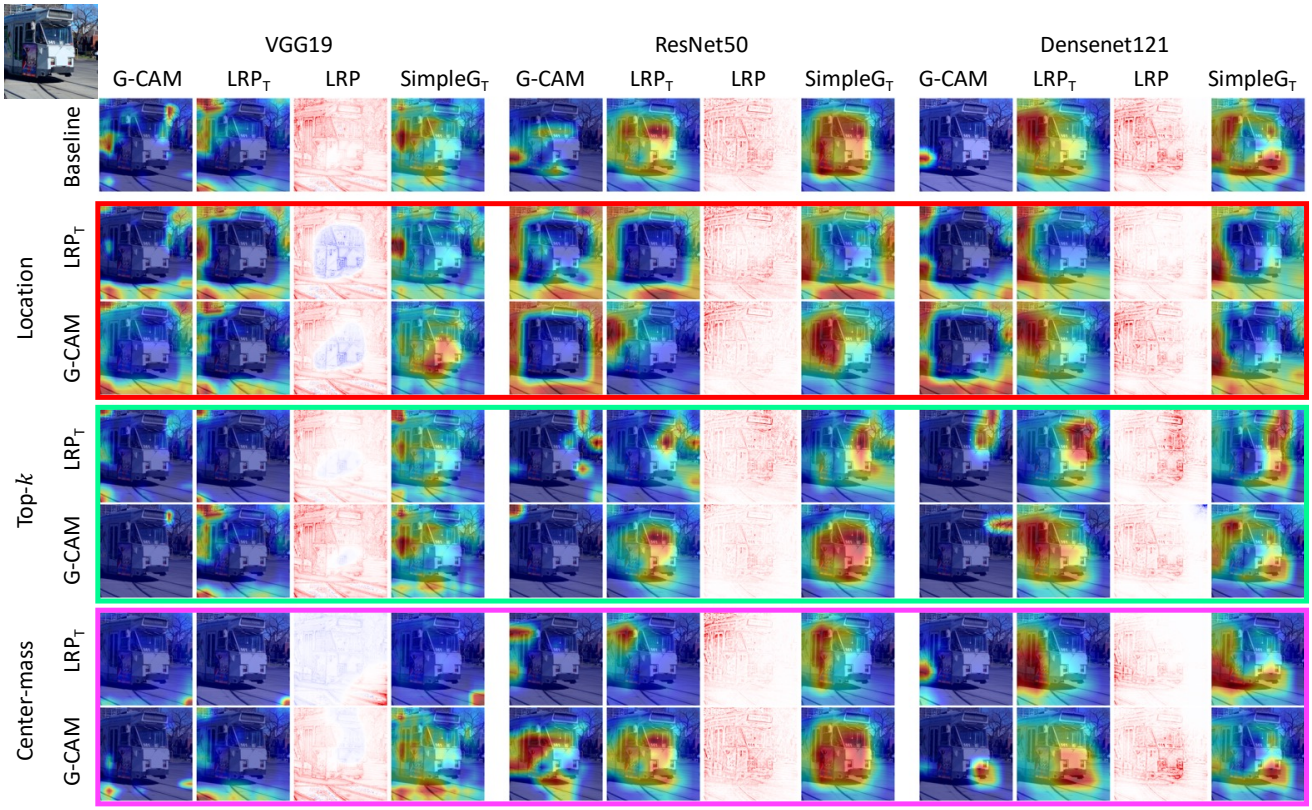


Figure 2. Interpretations of the baseline (pre-trained) and the passive fooled models on an image from ImageNet validation set, of which the class label is *'Streetcar'* (shown in top-left corner). **The topmost row** shows the baseline interpretations for three representative models, VGG19, ResNet50 and DenseNet121 by Grad-CAM, LRP$_T$, LRP and SimpleG$_T$ given the true class, 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 baseline explanation results are changed dramatically when fooled with each interpreter and fooling type. The transferability among methods should only be compared within each model architecture and fooling type.

$s_i(c, c') = (\mathbf{h}_c^{\mathcal{I}}(\boldsymbol{w}^*_{\text{fool}}), \mathbf{h}_{c'}^{\mathcal{I}}(\boldsymbol{w}_0))$ as the Spearman rank correlation [1] between the two heatmaps for $\mathbf{x}_i$, generated with $\mathcal{I}$. Then, define $t_i(\boldsymbol{w}_0, \boldsymbol{w}^*_{\text{fool}}, \mathcal{I}) = s_i(c_1, c_2) - s_i(c_1, c_1)$ for getting explanations with c_1 and $t_i(\boldsymbol{w}_0, \boldsymbol{w}^*_{\text{fool}}, \mathcal{I}) = s_i(c_1, c_2) - s_i(c_2, c_2)$ for c_2.

With above test losses for each fooling method f, the FSR for f and the interpreter $\mathcal{I}$ is defined as

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

in which $\mathbf{1}(\cdot)$ is an indicator function and R_f is a predefined interval for each fooling method. The meaning of R_f is a threshold for determining whether interpretations are fooled successfully. 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. Figure 3 shows how we set R_f empirically for the Passive folling. Note that the criterion is determined conservatively to ensure the qualitative success of the fooling. In short, the higher the FSR metric is, the more successful f is for the interpreter $\mathcal{I}$.

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

4.3. Passive and Active fooling results

In Figure 2 and Table 2, we present qualitative and quantitative results regarding our three Passive foolings. 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

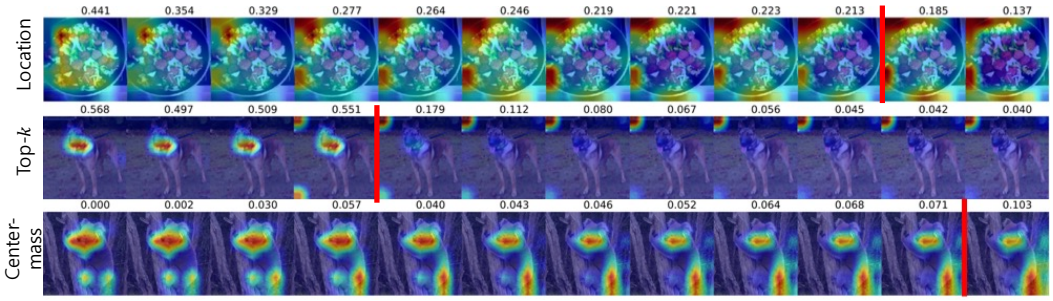


Figure 3. Visualizations and test lossed during Passive fooling training. The numbers above figure are the test losses defined in the manuscript for each fooling type. The red lines stand for the threshold of the success and failure of the fooling determined based on the visualization.

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 2. 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 is fooled or not. The transferability among methods should only be compared within the model and fooling type.

in the center, comparing $(1, 5)$ and $(3, 5)$ in Figure 2 for example [1]. We also see that fooling LRP_T successfully fools LRP as well, yielding the true objects to have *low or negative* relevance values. 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 baseline and middle green colored box in Figure 2. For the Center-mass fooling, the center of the heatmaps are now altered to the meaningless part of the images, yielding completely different interpretation 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 2. We can observe that all FSR of fooling methods are higher than 50% for the matched cases (bold underlined), except for the Location fooling on DenseNet121.

Next, for the Active fooling, from the qualitative results in Figure 4 and the quantitative results in Table 3, 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 are not clearly swapped, they tend to spread out to both c_1 and c_2 objects, which becomes less informative; compare between the $(1, 9)$ and $(2, 9)$ 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 mostly successful on VGG19 and ResNet50. For the case of DenseNet121, however, the FSR values are almost 0, which suggests the hardness of active fooling for DenseNet121. Such discrepancy for DenseNet may be also partly due to the conservative threshold we used for computing FSR since the visualization in Figure 4 shows some meaningful fooling also happens for DenseNet121 as well.

The significance of above results lie in the fact that the classification accuracies of all manipulated models are around the same as that of the original models shown in Table 1! Furthermore, we show in Table 4 that the decreases of Top-5 accuracies in Active fooling is not focused in the c_1 and c_2 classes, but spread out to the whole 1000 classes. Note our model manipulation affects the *entire* validation dataset without any access to it, unlike the common adversarial attack which has access to each input data point [9].

Importantly, we also emphasize that our fooling one interpretation method is *transferable* to other interpretation

[1] (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$_T$	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 3. 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.

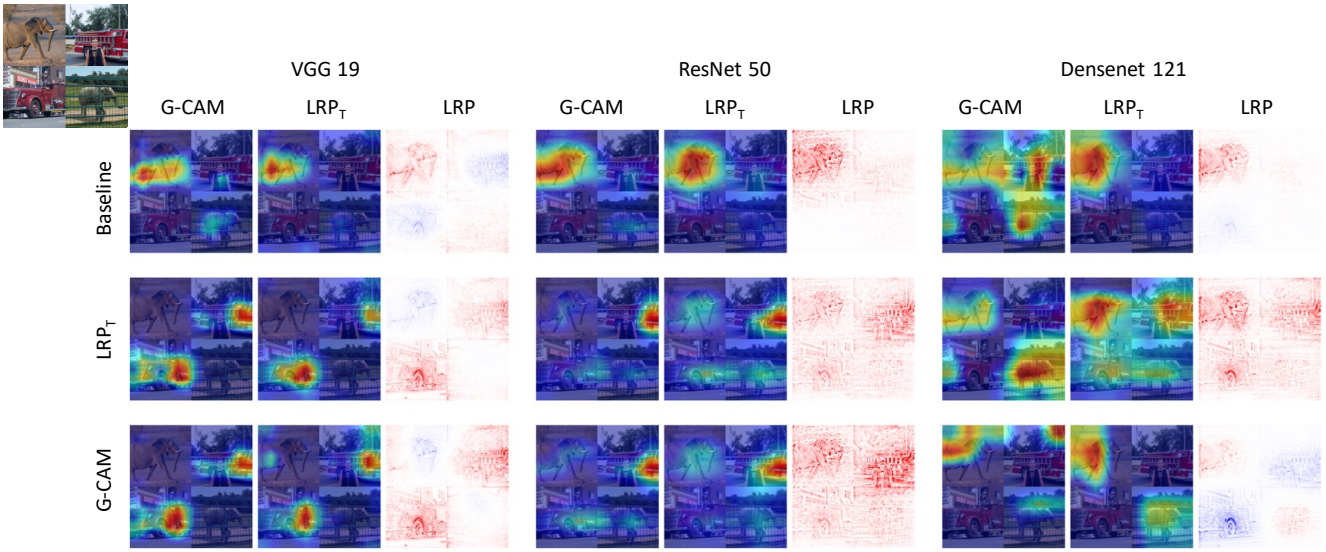


Figure 4. Explanations of original and active fooled models for c_1="African Elephant" from synthetic test images, which contain both Elephant and Firetruck (c_2) in different parts of the images for class c_1. **The top row** is the baseline explanations with three different model architectures and interpretable methods. **The middle** and **bottom row** are the explanations for the actively fooled models using Grad-CAM and LRP$_T$, respectively. We can see that the explanations for c_1 highlight c_2 instead of c_1. Transferability among methods should only be compared within each architecture.

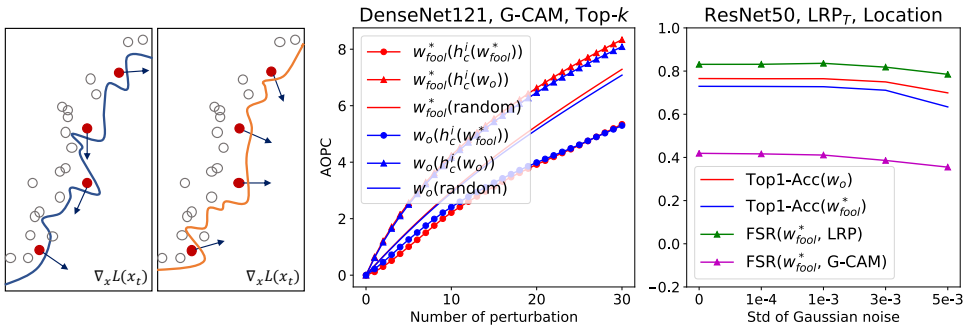
Models	VGG19		ResNet50		DenseNet121	
Accuracy (%)	c_1	c_2	c_1	c_2	c_1	c_2
Baseline	98.0	94.0	100.0	88.0	98.0	90.0
LRP$_T$	96.0	78.0	98.0	94.0	100.0	90.0
Grad-CAM	98.0	96.0	100.0	80.0	98.0	94.0

Table 4. Test accuracies on ImageNet validation set for c_1 and c_2 classes, when the model is Active fooled from c_1 to c_2. Each class has 50 images. Note that accuracies of c_1 and c_2 are quite similar to the baseline after the fooling, considering the size of test set. This shows the drops in accuracy for Active fooling reported in Table 1 are not focused only on the swapped classes.

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

Here, we try to give intuition why the fooling of interpretations via our model manipulation occurs. Motivated by [9], we simplify the problem by examining how the *gradient of inputs* changes as the model gets manipulated, instead of explaining with each interpretation method. Note the interpretation methods we used employ formula that are related to some form of gradients; SimpleG is using the gradient of the input, Grad-CAM is a function of the gradient of the representation at certain layer, and LRP turns out to be similar to gradient times inputs [15]. As illustrated in Figure 5(a), the same test data can be classified with almost the same accuracy even if the decision boundaries are somewhat



(a) Decision boundary (b) AOPC of original and fooled model (c) Gaussian noise addition on weight

Figure 5. (a) Two possible decision boundaries with the same accuracy, (b) AOPC of original and fooled model by Top-k with DenseNet121, Grad-CAM and (c) Robustness of Location fooling to Gaussian noise perturbation on weight parameters with ResNet50, LRP$_T$

changed. However, the gradients of inputs can be altered a lot, which may lead to completely different interpretations. Hence, what our manipulation step undertake is to fine-tune the model to have a certain decision boundary that make the gradients point as we wanted, while keeping the ability to classify the data.

The shared properties among the three interpretation methods, namely, the gradient information, also provide intuition on why transferability of fooling among them exists. Because we believe the inner information of decision making process they capture is quite similar, the decision boundaries for fooling one interpretation method can give similar wrong feature importance to the other methods as well. However, the asymmetry of transferability among three interpretations should be analyzed further. Furthermore, as also mentioned in Section 4, the level of fooling seems to have intriguing connection with the model complexity; particularly for Active fooling, the level of success is in the order of VGG19, Resnet50, and Densenet121, the same order of the model complexity. This resembles the similar finding in [10], which argues the model complexity is the key attribute on the success rate of adversarial attacks and the transferability.

One may argue that the fooling of the explanations could result in the actual change of the model, i.e., model focusing on the altered highlighted regions for making the prediction. To that regard, we employ Area Over Prediction Curve (AOPC) [27], which is a principled way of quantitatively evaluating the validity of neural network interpretations, to check whether the manipulated model also has been significantly changed by fooling the interpretation. From the definition of AOPC, we can conclude that the interpretation is closely tied to the actual prediction procedure of the model if AOPC rapidly increases with the number of perturbation of input pixels. Figure 5(b) shows the average AOPC curves on 10K validation images for the original and manipulated DenseNet121 models, $\boldsymbol{w}_o$ and $\boldsymbol{w}^*_{\text{fool}}$, with three different perturbation orders; i.e., with respect to the $\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}_o)$ scores, $\mathbf{h}^{\mathcal{I}}_c(\boldsymbol{w}^*_{\text{fool}})$ scores, and a random order. We did the Top-k fooling with $\mathcal{I}$ being Grad-CAM. 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, 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.

Another discussion point one can raise is about the *robustness* of our fooling. We claim that detecting or undoing our model manipulation would not be easy as we cannot easily access the original interpretation results or training data. Figure 5(c) shows a feasible attempt, inspired by Roth et al. [25], fails for detecting our manipulation. Namely, as the adversarial input examples can be detected by adding small Gaussian perturbation to the input,we may also suspect that adding small Gaussian noise to the parameters might reveal our fooling. However, Figure 5(c) shows that $\boldsymbol{w}_o$ and $\boldsymbol{w}^*_{\text{fool}}$ behave very similarly in terms of Top-1 accuracy as we increase the Gaussian noise level, and FSR do not change radically, either.

We believe this paper can open another research venue regarding designing more robust interpretation methods. We argue that checking robustness of interpretation methods with respect to our adversarial manipulation should be an indispensable criterion for the interpreters in addition to the sanity checks proposed in [2], since the Grad-CAM pass 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, more effective algorithms for model manipulation and defining more concrete metrics that check the stability of the interpretation methods could be another topics to consider.

References

[1] *Spearman Rank Correlation Coefficient*, pages 502–505. Springer New York, New York, NY, 2008.

[2] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity chekcs for saliency maps. In *NeurIPS*, 2018.

[3] D. Alvares-Melis and T. S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*, 2018.

[4] A. Arbor. Elephant pulls fire truck at the franzen brothers circus, 1996.

[5] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

[6] 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.

[7] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. In *arXiv:1702.08608v2*, 2017.

[8] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference (ACM)*, pages 214–226, 2012.

[9] A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. In *AAAI*, 2019.

[10] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arxiv preprint (2014), 2014.

[11] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models. In *https://arxiv.org/pdf/1802.01933.pdf*, 2018.

[12] D. Gunning. Explainable artificial intelligence (XAI). In *Defense Advanced Research Projects Agency (DARPA)*, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[15] e. a. Kindermans, Pieter-Jan. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016.

[16] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (un) reliability of saliency methods. *https://arxiv.org/pdf/1711.00867.pdf*, 2017.

[17] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[18] S. Lapuschkin, A. Binder, K.-R. Müller, and W. Samek. Understanding and comparing deep neural networks for age and gender classification. In *ICCVW*, 2017.

[19] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

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

[21] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[23] V. Petsiuk, A. Das, and K. Saenko. RISE: Randomized input sampling for explanation of black-box models. In *BMVC*, 2018.

[24] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *KDD*, 2016.

[25] K. Roth, Y. Kilcher, and T. Hofmann. The odds are odd: A statistical test for detecting adversarial examples. *arXiv preprint arXiv:1902.04818*, 2019.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[27] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[28] W. Samek, G. Montavon, and K.-R. Müller. Interpreting and explaining deep models in computer vision. In *CVPR Tutorial (http://interpretable-ml.org/cvpr2018tutorial/)*, 2018.

[29] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization. 2017.

[30] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017.

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

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[33] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR Workshop*, 2015.

[34] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

[35] B. Yu. Stability. *Bernoulli*, 19(4):1484–1500, 2013.

[36] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[37] X. Zhang, N. Wang, S. Ji, H. Shen, and T. Wang. Interpretable deep learning under fire. *arXiv preprint arXiv:1812.00891*, 2019.