# Learning Propagation Rules
# for Attribution Map Generation

Yiding Yang[1], Jiayan Qiu[2], Mingli Song[3], Dacheng Tao[2], and Xinchao Wang[1]

[1] Stevens Institute of Technology, Hoboken, NJ 07030, USA
{yyang99,xinchao.wang}@stevens.edu
[2] UBTECH Sydney AI Centre, School of Computer Science, Faculty of Engineering,
The University of Sydney, Darlington, NSW 2008, Australia
{jqiu3225@uni.sydney.edu.au,dacheng.tao@sydney.edu.au}
[3] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
brooksong@zju.edu.cn

**Abstract.** Prior gradient-based attribution-map methods rely on hand-crafted propagation rules for the non-linear/activation layers during the backward pass, so as to produce gradients of the input and then the attribution map. Despite the promising results achieved, such methods are sensitive to the non-informative high-frequency components and lack adaptability for various models and samples. In this paper, we propose a dedicated method to generate attribution maps that allow us to learn the propagation rules automatically, overcoming the flaws of the hand-crafted ones. Specifically, we introduce a learnable plugin module, which enables adaptive propagation rules for each pixel, to the non-linear layers during the backward pass for mask generating. The masked input image is then fed into the model again to obtain new output that can be used as a guidance when combined with the original one. The introduced learnable module can be trained under any auto-grad framework with higher-order differential support. As demonstrated on five datasets and six network architectures, the proposed method yields state-of-the-art results and gives cleaner and more visually plausible attribution maps.

**Keywords:** Propagation Rules, Attributions Maps, Learnable Module

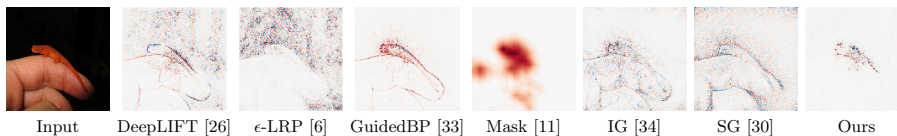| Input | DeepLIFT [26] | $\epsilon$-LRP [6] | GuidedBP [33] | Mask [11] | IG [34] | SG [30] | Ours |

Fig. 1: What makes this image a *newt*? This figure shows the attribution maps generated by different methods. Existing gradient-based methods fail even in this simple case. For example, IG and GuidedBP focus on non-relevant regions, such as the boundary of the hand. Our method, on the other hand, produces cleaner and more focused attribution map.
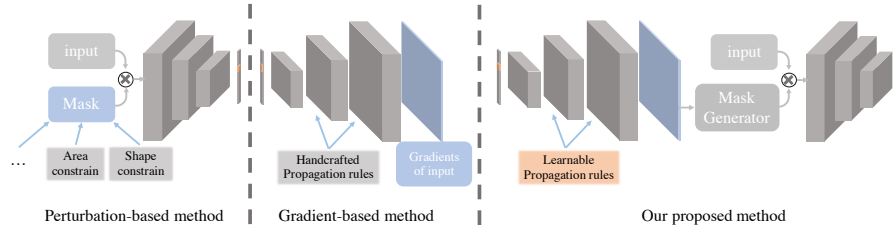
Fig. 2: Comparing the perturbation-based method, graident-based method, and our proposed method for generating attribution maps. Different from the perturbation-based method that introduces various constraints to the mask, we generate the mask by making use of the gradients of the input; Moreover, unlike the gradient-based method that handcrafts the propagation rules, we make them learnable.

## 1   Introduction

Deep learning has made encouraging progress and yielded state-of-the-art performances in almost all vision and language tasks. The gratifying results, however, come at cost of huge amount of training effort as well as the often uninterpretable behaviors, making deep networks less dependable under some circumstances such as medical image processing. Recently, interpreting deep networks has aroused more and more attention from researchers [1, 4, 22, 42, 12, 2, 9, 40, 7, 36, 37, 25, 39, 38]. Among the many endeavors, estimating the *attribution map* has become a mainstream direction. The main goal of producing an attribution map is to generate a mapping between the pixels and their corresponding contributions to the prediction, so that the supports of the prediction can be discovered. The work of [31, 32] have also demonstrated that attention maps can be utilized in estimating task transferability.

Existing attribution-map generation methods can be divided into three categories: optimization-based, perturbation-based, and gradient-based methods. Optimization-based methods produce attribution maps using conventional optimization methods like signal estimation [16], and local function approximation [21]. Such optimizers, however, often require a large number of samples, making them data-dependent and time-consuming. Perturbation-based methods, on the other hand, produce attribution maps by modifying the input image according to a mask and then recording the change of output. However, they ignore the original gradients of the input. Gradient-based methods explicitly utilize gradients of the input for attribution map generation, and therefore encode the interaction across different pixels, yielding more informative attribution maps [19]. Given a trained model with fixed parameters, gradients of the input are obtained through loss back-propagation, where existing methods focus on designing hand-crafted propagation rules for the non-linear/activation layers [6, 3]. However, such pre-defined and thus fixed rules lack adaptability for various models and samples.

In this paper, we propose a new method to generate the attribution map that makes the propagation rules for the non-linear layers *learnable*, and optimize the rules using supervision from the model and the input image themselves. In Fig. 1, we compare our produced attribution map with those obtained from the state-of-the-art methods. Conventional gradient-based methods such as DeepLIFT and $\epsilon$-LRP are prone to noisy attributions even for the uniformed-colored background. Most of the methods focus on non-relevant high-frequency regions, such as the boundary of the hand. Our method, thanks to the more flexible rules, generates a neat and more focused attribution map. Fig. 2 illustrates a comparison of different methods. Unlike conventional gradient-based approaches that rely on a single unifying hand-crafted propagation rule for all models and samples, we now make the rules adaptive for any given sample and model. Specifically, within our method, each output feature of the non-linear layers is allowed to behave differently during the backward pass, making it possible to learn more flexible and advisable propagation rules for the attribution map. As a result, the non-informative regions, e.g. the high-frequency ones, are suppressed under the supervision, leading to a cleaner attribution map.

To learn the propagation rules, a new optimization scheme is proposed as shown in Fig. 3. The input image is first fed into a trained neural network with fixed parameters to obtain the original prediction. Then, gradients of the input can be obtained during the backward pass, in which process, a *learnable plugin module*, e.g. neural network, is introduced to control the propagation rules of the non-linear layers. After obtained the gradients of the input, we compute the attribution map and then generate masks for the input. The input image will then be masked and fed into the trained network for deriving the difference with respect to the original prediction. Such difference is adopted as a supervision to optimize the learnable plugin module through a new backward pass. Since the computation of second-order gradients is required, an auto-grad framework with higher-order differential support is used to implement the proposed optimization scheme.

Our contribution is therefore, to our best knowledge, the first dedicated approach that enables the learning of the propagation rules for the non-linear layers to generate attribution maps. Unlike the hand-crafted rules, our method makes it possible to find adaptive propagation rules for any given model and sample. The learning of the rule is achieved via a novel optimization scheme: the learnable module we introduced can be optimized under any auto-grad framework with higher-order differential support. We conduct experiments on three different datasets and six models with different architectures. Our proposed method yields state-of-the-art results and produces a cleaner attribution map.

## 2   Related Work

Here we give a brief description of the related work. We start by reviewing the attribution-map methods of three categories: optimization-based, gradient-based, and perturbation-based methods. We then discuss the higher-order differ-

ential algorithms for implementing our proposed method. Note that the proposed method differs from all three categories of attribution-map methods. Specifically, compared with optimization-based methods, our proposed method depends only on given samples; compared with gradient-based methods, our proposed method enables the learning of the propagation rules; compared with perturbation-based methods, our proposed method involves the gradients of inputs as the condition to generate the mask rather than the human designed constraints.

**Optimization-based methods.**  These methods adopt conventional optimization scheme to generate an attribution map. For example, PatternNet [16] designs a signal detector to filter out the non-informative components. Then, a quality measurement criterion is introduced to optimize the attribution map generation. Instance feature selector [8] learns a feature selector by maximizing the mutual information between the selected features and the model's response. Another method, LIME [21], locally approximates a non-linear model with a linear function on the given sample, and then generates the attribution map from the linear function. However, these methods are data-dependent and time-consuming.

**Gradient-based methods.** Such methods utilize gradients of the input to generate an attribution map. For example, Deep saliency [28] generates the attribution map by backwarding the loss with respect to the input and taking the absolute value of the gradients. Moreover, the element-wise multiplication between the input and its gradients improves the performance [27]. Another method, Guided backpropagation [33], shows that ignoring the negative gradients helps to distinguish the contribution of each pixel. As for DeepLIFT [26], reference features are added to the non-linear layers to reduce the influence from baseline. However, the fixed propagation rules of these methods lack adaptability for various models and samples.

**Perturbation-based methods.** Methods along this line make the assumption that removing important pixels will degrade the prediction accuracy, and generate the mask based on some constraints. One of the methods, Occlusion [41], generates the attribution map by systematically occluding different parts of the input image and then recording the change of output. Moreover, it is also possible to obtain an occlusion mask by learning [11] rather than brute force searching. sMask [10] introduce predefined area constrain and smooth constrain.

**Higher-order differential algorithms.** High-order differential algorithms make the second-order gradients computation possible [13, 18], and are thus essential for our proposed attribution-map method. Most of the current deep learning libraries implement these algorithms. In Pytorch, for example, gradients of a variable remains a variable, which enables the computation of higher-order gradients by recursively computing the first-order gradients. These implementations serve as the backbone of our proposed optimization scheme.
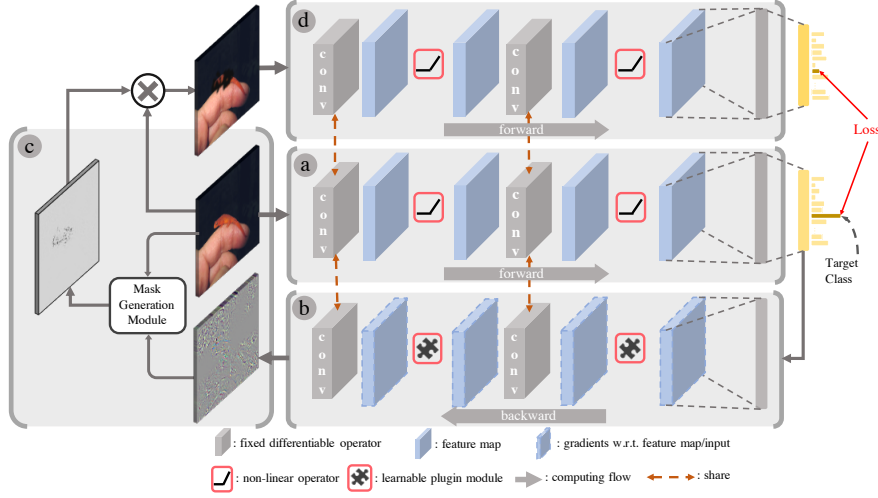
Fig. 3: Illustration of our proposed optimization scheme. Step *a* forwards the input image through the trained model to obtain the original activation and its loss. Specifically, we use the pre-softmax/pre-sigmoid output as the activation of the model. Step *b* then backwards the gradients of input image with respect to the prediction loss, in which we introduce the *Learnable Plugin Module*. The attribution map is obtained by the element-wise multiplication between the input image and its gradients. Next, step *c* generates the masks with the attribution map. Finally, step *d* masks the input image and then forwards the masked image through the trained model to get a new activation. The difference between the new activation and the original one of the target class serves as the loss, which is used to optimize the learnable plugin module.

## 3   Method

Propagation rules for the non-linear layers during the backward pass can be treated as a pixel-to-pixel mapping between gradients of the adjacent feature maps. For gradient-based attribution-map methods, they are proven to be unified and distinguished only on the propagation rules of the non-linear layers [3]. However, existing gradient-based methods formulate the gradient computation as hand-crafted rules, which are hard to fit all models and samples. For example, a method may work well for images with clean backgrounds but fail on those with complex backgrounds.

To this end, we propose a novel method, which makes the rules learnable. The rules will be optimized individually under the supervision of every combination of the input image and the trained neural network, making it adaptive for given models and samples. Fig. 3 illustrates the proposed optimization scheme, which is also based on the gradient-descent optimization method. There are four steps in a single optimization iteration for attribution map generation. In Step 1, we forward the input image throughout the network; in Step 2, we backward gradients of the input through our learnable plugin module for attribution map

generation; in Step 3, we generate the mask with the attribution map; in Step 4, we forward the masked input image to obtain the loss by computing the difference of activations between the original input and the masked one, where the activation is referred as the pre-softmax/pre-sigmoid output of the model.

### 3.1    Step 1: Forwarding the input image

In this step, the input image is fed into a trained neural network model to obtain the activation. Once the target class to be interpreted is chosen, we set the gradient of the activation to one for target class and zero for the others which is commonly used in gradient-based methods [33, 27, 6]. We then pass the gradient to the next step.

### 3.2    Step 2: Backwarding the gradients of the input

Given the gradient of the model's activation from previous step, the gradients of the input image can be computed through the back-propagation algorithm. During this process, we implement our proposed learnable plugin module to learn the propagation rules for non-linear layers, instead of modifying the hand-crafted function.

Specifically, after feeding a feature map $f_{in}$ into a non-linear function $g$, we will have the output $f_{out} = g(f_{in})$. Let $\mathcal{L}$ denotes the training loss during the backward pass. The gradient of $f_{out}$ is the partial derivatives of $\mathcal{L}$ with respect to the output feature map that be expressed as $\frac{\partial \mathcal{L}}{\partial f_{out}}$.

Then, following the chain rule of the back-propagation algorithm, the partial derivatives of $\mathcal{L}$ with respect to $f_{out}$, can be computed as

$$\frac{\partial \mathcal{L}}{\partial f_{in}} = \frac{\partial \mathcal{L}}{\partial f_{out}} \cdot \frac{\partial g(x)}{\partial f_{in}} = \frac{\partial \mathcal{L}}{\partial f_{out}} \cdot \frac{\partial g(f_{in})}{\partial f_{in}}, \tag{1}$$

where $\frac{\partial g(x)}{\partial x}$ denotes the derivation of non-linear function $g$ with respect to it's input.

In existing gradient-based methods, $\frac{\partial g(x)}{\partial x}$ is manually modified for different purposes, such as ignoring the neurons that suppress the target output [33]. Although numerous hand-crafted propagation rules are proposed, none of them is optimal for all scenarios. For example, some hand-crafted rules are unresponsive to certain target class, while others may be too sensitive to ignore non-relevant high-frequency components.

Therefore, instead of using a fixed hand-crafted $\frac{\partial g(x)}{\partial x}$, we introduce a learnable plugin module, denoted as $G$, as the basic modules. This module takes the gradients of feature map $\frac{\partial \mathcal{L}}{\partial f_{out}}$ from the upper layer and computes $\frac{\partial \mathcal{L}}{\partial f_{in}}$ as

$$\frac{\partial \mathcal{L}}{\partial f_{in}} = G(\frac{\partial \mathcal{L}}{\partial f_{out}}) \tag{2}$$

where $G$ can be plug-and-play without modifying the original architecture of the given trained neural network. We provide two architectures for $G$, For the

first architecture, we have $C$ parameters per layer, where $C$ is the number of channels. The operation of $G$ is similar to a standard convolutional operation without the sum operation, which shares parameters across different positions within a same layer. For the second architecture, we do not share parameters across different positions, leading to more flexible control of the rules at cost of more parameters. Once the gradients of the input are obtained, the attribution map can be computed as

$$\mathcal{A} = \frac{\partial \mathcal{L}}{\partial I} \circ I, \tag{3}$$

where $\mathcal{A}$, $I$, and $\frac{\partial \mathcal{L}}{\partial I}$ denote the attribution map, input image, its gradients respectively, and $\circ$ is Hadamard product. The pros and cons of multiplying the generated gradient with the input have been discussed in [30]. In this paper, this multiplication is adopted across all experiments.

### 3.3    Step 3: Mask generation

Given the attribution map, a mask can be generated to segment out the image parts that contribute most to the target-class recognition. Since the distribution of attribution map varies a lot, we propose a *Mask Generation Module* for generating suitable masks here.

It can be seen from Fig. 4 that the attribution map is first scaled to $[0, 1]$ and then shifted to a fixed center. Finally, we implement the mask generation using a sigmoid function. We write,

$$\mathcal{M}^p = 1 - \frac{1}{1 + e^{(-\gamma*(\mathcal{A}-\alpha))}}, \tag{4}$$

$$\mathcal{M}^n = \frac{1}{1 + e^{(-\gamma*(\mathcal{A}-\beta))}}, \tag{5}$$

where $\mathcal{M}^p$ denotes the positive mask that segments pixels with positive contribution, $\mathcal{M}^n$ denotes the negative mask that segments pixels with negative contribution, $\alpha$, $\beta$ denote the fixed centers and $\gamma$ denotes the scale factor for sharpening the mask.

Notice that many other mask generation strategies can be directly adopted here, leading to different properties of the generated attribution map. For example, in order to generate smoother mask, a Gaussian smooth function can be replaced for the previous mask generation functions, which can be written as:

$$\mathcal{M}^p = \frac{\sum_{v \in \mathcal{A}} \mathcal{S}_\sigma(u - v)\mathcal{A}(v)}{\sum_{v \in \mathcal{A}} \mathcal{S}_\sigma(u - v)}, \mathcal{S}_\sigma(u) = e^{-\frac{\|u\|^2}{2\sigma^2}} \tag{6}$$

where $\mathcal{A}$ is the generation attribution map, $u, v$ is the index of values in $\mathcal{A}$ and $\mathcal{S}$ is the smooth function.
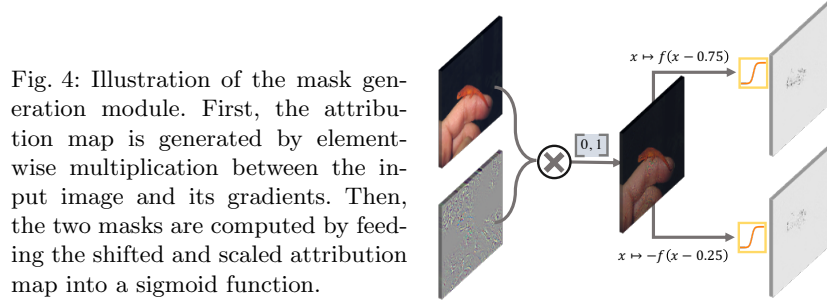
Fig. 4: Illustration of the mask generation module. First, the attribution map is generated by element-wise multiplication between the input image and its gradients. Then, the two masks are computed by feeding the shifted and scaled attribution map into a sigmoid function.

### 3.4 Step 4: Forwarding the masked image

Based on the generated masks, pixels with special contributions will be segmented out from the input image to form the masked image. We then forward the masked image through the trained model to obtain the difference between the activation of input image and masked image, which is used as the loss to train the learnable plugin module.

Some pixels in the masked image contribute to correct prediction while some pixels degrade the accuracy. Therefore, instead of measuring the contribution according to the sign of gradients, we propose a *sign-aware loss* to distinguish the pixels.

There are two terms in the sign-aware loss, a positive term and a negative one. The positive term $\mathcal{L}^p$ considers the positively contributed components of the input image:

$$\mathcal{L}^p = F_t(I \circ \mathcal{M}^p) - F_t(I), \tag{7}$$

where $F$ denotes the trained model, $t$ denotes the index of the target class, and $F_t(I)$ denotes the predicted possibility of $I$ to be class $t$. By the same token, the negative term is defined as

$$\mathcal{L}^n = F_t(I) - F_t(I \circ \mathcal{M}^n). \tag{8}$$

This term is defined as the difference of predictions between the input image and the negatively masked image, because deleting the negatively contributed pixels should improve the prediction accuracy.

To avoid the trivial solutions like an all-zero mask or an all-one mask, we introduce mask loss to constrain the strength of the generated masks:

$$\mathcal{L}^m = |\mathbf{1} - \mathcal{M}^p| + |\mathbf{1} - \mathcal{M}^n|, \tag{9}$$

where $| \bullet |$ denotes the $L_1$-Norm, $\mathbf{1}$ denotes matrix with all ones. We thus have the final loss function:

$$\mathcal{L} = (\mathcal{L}^p + \mathcal{L}^n) + \lambda \cdot \mathcal{L}^m, \tag{10}$$

where $\lambda$ is the hyper-parameter for loss balancing.

## 4    Experiments

### 4.1    Evaluation protocols

There are two kinds of attribution-map errors: error from the attribution-map method itself and error from the trained model. We conduct objective evaluation which focuses purely on the first type of error. We adopt most important relevant features (MoRF) curve as one of the metrics, as done by many other methods [23, 5, 6]. Specifically, we first sort the pixels in an ascending manner according to the attribution map, and then obtain the MoRF curve by incrementally computing the correlation between the different ratios of the activation and the ratios of masked pixels. We also adopt least important relevant features (LeRF) curve as one objective evaluation metric. LeRF is of the same setting as MoRF except it first sorts the pixels in a descending manner. For all objective evaluations, we set the upper limit of the number of masked pixels to be 5% of the entire input image. We derive the MoRF curve by averaging 1,000 random samples for stability.

ROAR [15] is another metric to evaluate the performance of attribution map. ROAR first replaces fraction of the pixels that are estimated by the attribution map as the most important ones with uninformative value. Then, the modified data are used to retrain the same model from scratch and test it on the modified test set. It claims that a good attribution map should lead to a sharp degradation of the performance on the modified dataset.

### 4.2    Implementation Details

**Attribution-map framework.** We build a PyTorch-based attribution-map toolbox, which implements the proposed optimization scheme and some of the compared methods. In our toolbox, the gradient-based methods are unified by sharing the forward and backward hook functions. In the objective experiments, since our only concern is about the model interpretation, the class with the highest prediction possibility is set as the ground truth. We then adopt the negative log likelihood function as the loss function. The running time of our method for an ImageNet-like input with a VGG-16 model is about 3s using an Nvidia 1080Ti GPU.

**Learnable plugin module.** For the objective experiments, we use the second architecture of *learnable plugin module*. Specifically, we set the parameter matrices of learnable plugin module to be of the same size as the input feature map. For every non-linear layers in the model, learnable plugin module first computes the Hadamard product between its parameter matrix and the input feature map, then follows a tanh activation function, We also conduct some experiments of the first archtecture of the learnable plugin module which is similar as the convolution without summation and shares parameters within one layer. Although the structure is concise, it leads to a significant improvement of the performance. The learnable plugin module is optimized with Adam [17]. Similar

to sMask [10], we train the plugin module separately for each sample. The plugin modules within different layers do not share parameters and are placed in every nonlinear layer within two convolution layers.

**Reference baseline.** In order to improve the flexibility of learnable plugin module, we adopt the reference baseline from DeepLIFT [26]. Specifically, an additional all-zero input is added in the forward pass to obtain the reference feature maps. Then, all the original features are modified by subtracting their corresponding reference feature map.

**Hyper-parameters settings.** We adopt same hyper-parameters to all experiments, with $\lambda$ set to be 0.1, $\alpha$ set to be 0.75, $\beta$ set to be 0.25, and $\gamma$ set to be 10. For the Adam optimizer, we set the learning rate to be 0.2. No weight decay is used. The performance with respect to these hyper-parameters are stable within a large value range, and the analysis will be presented in the sensitivity analysis section.

### 4.3   Compared Methods

Here, we give a brief description of the compared methods.

- **Gradient-based methods**. GradientXInput (GradXIn) [27] and DeepSaliency (Saliency) [28] generate attribution maps from the gradients of the input. Specifically, DeepSaliency ultilizes the gradients only, while GradXIn uses both the input and its gradients. $\epsilon$-LRP [5], and DeepLIFT [26], on the other hand, focus on designing hand-crafted fixed propagation rules to enhance performances. SQ_SG [15] is an improvement over smooth grad method by averaging the squared gradients. Integrated Gradients (IG) [34] and Smooth Gradients (SG) [30] compute the average gradients of multiple inputs by introducing integration paths and adding noises, respectively.
- **Perturbation-based methods**. Mask [11] treats the attribution map as a mask and learns it from a designed framework. sMask [10] adds more constraints to the mask. RISE [20] generates random masks and obtains the attribution map by linear combining these masks according to outputs of masked images.

The Mask method is tested on ImageNet dataset only, because it is designed for ImageNet-similar images. The hyper-parameters of Mask are set according to their published work. For the SmoothGrad method, the number of random noised images is set to 50. As for the IntegratedGrad method, the number of integrated images along the integral path from zero baseline to the original input is set to 50. The tolerance of pointing game is set to 15 for all compared methods.

### 4.4   Experimental Results

In this section, we first analyze the comparison results between our proposed method and the compared methods in the aspects of MoRF, LeRF, and ROAR. Then, we present the case study, the sensitivity analysis of hyper-parameters, and the ablation study to completely evaluate our proposed method.
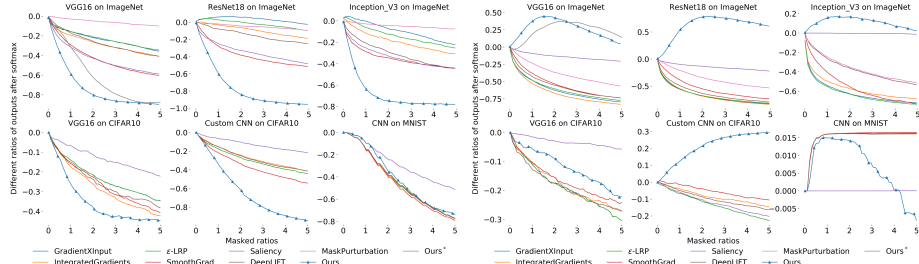
Fig. 5: MoRF (left) and LeRF (right) curves. The x-axis represents the masked ratio of the entire image and the y-axis represents the difference ratio of the activation after masking the input. Our method produces consistently steeper curves, especially for complex models and samples. Note that for the LeRF metric, our method provides the correct negative attributions, which will lead to an increase of the prediction accuracy when removing them. All other methods, as a comparison, fail and still generate positive attributions.
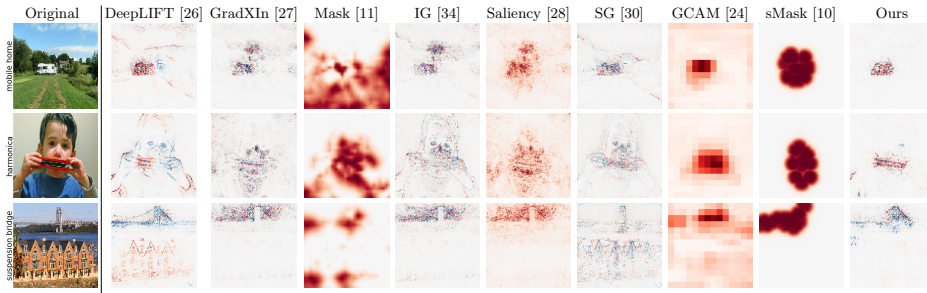


Fig. 6: Visualization of the attribution maps generated by different methods on ImageNet dataset. The attribution maps generated by our method are cleaner and more visually plausible. For gradient-based methods, some of them are too sensitive to ignore the non-relevant high-frequency components while others are not responsive enough to the target class (e.g. the last line). More visualization results will be presented in the supplemental material.

**MoRF/LeRF result analysis.** Fig. 5 compares the MoRF and LeRF curves of our proposed method and the compared methods. We conduct the objective experiments on three widely used datasets, MNIST, CIFAR-10, and ImageNet. ∗ means the plugin module with the first architecture which shares parameters across different positions. The analysis of the results on the three datasets is as follows:

– **MNIST** is a relative small and simple dataset, in which the images contain only unit digits with clean background. We test on it using a customized CNN model with two convolutional layers. It can be seen that all methods lead to similar performances on both the MoRF and LeRF curves. This can

be in part explained by the fact that, simple images cannot distinguish the potentials of these methods.

– **CIFAR-10** is a larger dataset, in which the images contain common objects but with low resolution. We test two CNN models on this dataset, a custom CNN model with four convolutional layers and a VGG-16 model [29]. It can be seen that our proposed method performs the best on both the MoRF and LeRF curves.

– **ImageNet** is one of the largest and most complex datasets, in which the images come from the real-world scenes. We test all methods with three state-of-the-art models on this dataset, a VGG-16 model, a ResNet18 model [14], and a Inception-V3 model [35]. It can be seen that our method performs the best on the MoRF curve consistently by a large margin. As for the LeRF cure, thanks to the sign-aware loss, our method gives the correct negative attributions while all compared methods fail.

Table 1: Evaluation of ROAR on the CI-FAR10 dataset with the custom CNN model. Our method consistantly performs better that others especially when the fraction of removed pixels are large. We report the test accuracy on the modifed dataset (lower is better).

| Fraction | Original | Random | SQ_SG | Ours |
|---|---|---|---|---|
| 40% | 80.73% | 73.65% | 74.90% | 71.83% |
| 50% | 80.73% | 72.46% | 72.97% | 69.81% |
| 60% | 80.73% | 70.89% | 70.79% | 67.32% |
| 70% | 80.73% | 68.98% | 68.33% | 63.80% |
| 80% | 80.73% | 65.85% | 65.19% | 59.08% |
| 90% | 80.73% | 59.58% | 57.47% | 50.72% |



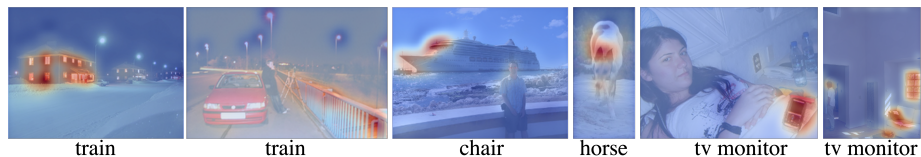train          train          chair          horse          tv monitor          tv monitor

Fig. 7: Illustration of some misclassified images and their corresponding attribution maps. The predictions of these images are train, train, chair, horse, tv monitor and tv monitor respectively. Red color highlights the supports of the predictions. Note that although the attribution map does not align well with the human perception, masking the image according to the attribution map still lead to a significant drop of the wrong prediction.

We visualize the attribution maps on ImageNet dataset with a trained VGG-16 model in Fig. 6. It can be seen that some compared methods are too sensitive to ignore the non-relevant high-frequency components (e.g. the grass and the boundary of trees). Other methods, unfortunately, fail to localize the most contributed areas to the prediction (e.g. the results in the last line). As a comparison, our method provides consistently cleaner and more focused attribution maps.

**ROAR result analysis.** Tab. 1 shows the ROAR result. We change the fraction of removed pixels from 40% to 90% and retrain the model on the modified dataset. The test accuracy is reported and lower is better. Our method consistantly leads to a lower accuracy on the modified test set.

**Case study.** We first conduct a case study of some miscalssfied images shown in Fig. 7. It can be seen that even the generated attribution map does not align well with the human perception, masking the input according to the attribution map stills lead to a significant drop of the wrong prediction. We also conduct a case study on a composite image. It contains two objects from different classes. Our method is truly responsive to the target class, as can be seen from Fig. 8, and focuses on the most informative areas while many gradient-based methods are not sensitive to the target class or even give the opposite signs (like DeepLIFT).
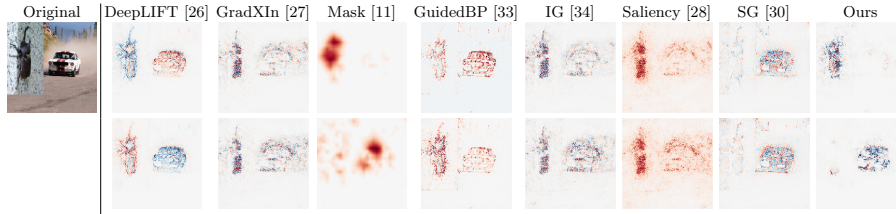


Fig. 8: Visualization of a composite image with two different objects. The target name of the first line is Rhinoceros beetle while the target name of the second line is Landrover. Many gradient-based methods are not sensitive to the target class.

**Sensitivity analysis.** Here, we conduct experiments for analyzing the sensitivity of the four hyper-parameters in our method including the scale factor $\gamma$, two fixed centers $\alpha$ and $\beta$, and $\lambda$ for loss balancing. All results are obtained using a trained VGG-16 model performing on the ImageNet dataset. In order to conduct a more comprehensive analysis, we also evaluate the sensitivity of learning rates and optimization iterations. The performance is measured by Area Under The Curve (AUC) of the MoRF curve, for which a lower value indicates a better result. We present all the results in Tab. 2, where intensities of the color are associated with the AUC values. It can be seen that the performance of our method is not sensitive to $\lambda$. For other hyper-parameters, the performance stays stable when they are set in a reasonable range.

**Ablation Studies.** We conduct ablation studies to analyze the effect of two terms, including the sign-aware loss and the reference baseline. The comparison results between the full model and models without one of two terms are presented in Fig. 9. It can be seen that the sign-aware loss improves the performance on LeRF curve by a significant margin with cost of a little affect on the performance of MoRF. The intuition behind is that the sign-aware loss, which

Table 2: Results of sensitivity analysis for the hyper-parameters. AUC is the area under MoRF curve and lower is better. The performance is not sensitive to $\lambda$ and stays stable when choosing reasonable values for other hyper-parameters.

| $\gamma$ | 1 | 7.5 | 14 | 21 | 27 | 34 | 40 | 47 | 53.5 | 60 |
|---|---|---|---|---|---|---|---|---|---|---|
| AUC | 0.40 | 0.27 | 0.16 | 0.13 | 0.14 | 0.15 | 0.17 | 0.18 | 0.20 | 0.22 |
| Iters | 1 | 8 | 14 | 21 | 27 | 34 | 40 | 47 | 54 | 60 |
| AUC | 0.86 | 0.29 | 0.27 | 0.27 | 0.26 | 0.24 | 0.23 | 0.23 | 0.22 | 0.23 |
| $\lambda$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
| AUC | 0.23 | 0.22 | 0.22 | 0.22 | 0.22 | 0.22 | 0.21 | 0.22 | 0.21 | 0.22 |
| lr | 0.01 | 0.06 | 0.12 | 0.17 | 0.23 | 0.28 | 0.34 | 0.39 | 0.45 | 0.5 |
| AUC | 0.40 | 0.33 | 0.29 | 0.26 | 0.25 | 0.25 | 0.21 | 0.21 | 0.20 | 0.21 |
| $\alpha$ | 0.55 | 0.59 | 0.64 | 0.68 | 0.73 | 0.77 | 0.82 | 0.86 | 0.90 | 0.95 |
| AUC | 0.47 | 0.37 | 0.30 | 0.26 | 0.23 | 0.21 | 0.19 | 0.18 | 0.18 | 0.21 |

Fig. 9: Ablation studies of the sign-aware loss and the reference baseline. The sign-aware loss dramatically improves the performance on LeRF curve while the baseline reference has a little influence on the performance.
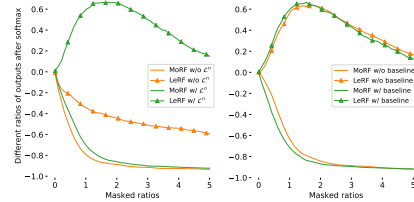
contains two branches, will generate the guided supervisions for two types of mask separately. As for the reference baseline, it influences the performance of the proposed method on both LeRF and MoRF curves slightly, implying that the learnable plugin module is already flexible enough even without the cues provided by the added reference features. We also tried to use the standard convolution as the plugin module but fail to generate a meaningful attribution map. This can be partially explained by that, the standard convolution operation does a substantial change to the gradients, leading to a pointless attribution map that is unrelated to the model anymore.

## 5 Conclusion

In this paper, we propose a dedicated attribution-map method that enables the propagation rules learnable for the non-linear layers, so as to overcome the drawbacks of existing gradient-base methods. The propagation rules are controlled by the plugin module and can be optimized by the proposed optimization scheme under any auto-grad framework with higher-order differential support. The learnable rules are adaptive, thanks to the supervision from the model and the input themselves. As demonstrated on several datasets and models, our method yields state-of-the-art results and produces cleaner and more focused attribution maps.

# References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: Advances in Neural Information Processing Systems. pp. 9505–9515 (2018)
2. Alvarez-Melis, D., Jaakkola, T.S.: Towards robust interpretability with self-explaining neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 7786–7795 (2018)
3. Ancona, M., Ceolini, E., Öztireli, C., Gross, M.: Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104 (2017)
4. Ancona, M., Öztireli, C., Gross, M.: Explaining deep neural networks with a polynomial time algorithm for shapley values approximation. arXiv preprint arXiv:1903.10992 (2019)
5. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
6. Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R., Samek, W.: Layer-wise relevance propagation for neural networks with local renormalization layers. In: International Conference on Artificial Neural Networks. pp. 63–71 (2016)
7. Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., Tian, Q.: Data-free learning of student networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3514–3522 (2019)
8. Chen, J., Song, L., Wainwright, M.J., Jordan, M.I.: Learning to explain: An information-theoretic perspective on model interpretation. arXiv preprint arXiv:1802.07814 (2018)
9. Feng, Z., Wang, X., Ke, C., Zeng, A.X., Tao, D., Song, M.: Dual swap disentangling. In: Advances in Neural Information Processing Systems 31 (2018)
10. Fong, R., Patrick, M., Vedaldi, A.: Understanding deep networks via extremal perturbations and smooth masks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2950–2958 (2019)
11. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3429–3437 (2017)
12. Ghorbani, A., Abid, A., Zou, J.: Interpretation of neural networks is fragile. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3681–3688 (2019)
13. Griewank, A., Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation, vol. 105. SIAM (2008)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Hooker, S., Erhan, D., Kindermans, P.J., Kim, B.: A benchmark for interpretability methods in deep neural networks. In: Advances in Neural Information Processing Systems. pp. 9737–9748 (2019)
16. Kindermans, P.J., Schütt, K.T., Alber, M., Müller, K.R., Erhan, D., Kim, B., Dähne, S.: Learning how to explain neural networks: Patternnet and patternattribution. arXiv preprint arXiv:1705.05598 (2017)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

18. Maclaurin, D.: Modeling, inference and optimization with composable differentiable procedures. Ph.D. thesis (2016)
19. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017)
20. Petsiuk, V., Das, A., Saenko, K.: Rise: Randomized input sampling for explanation of black-box models. arXiv preprint arXiv:1806.07421 (2018)
21. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
22. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
23. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R.: Evaluating the visualization of what a deep neural network has learned. IEEE transactions on neural networks and learning systems **28**(11), 2660–2673 (2016)
24. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Gradcam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 618–626 (2017)
25. Shen, C., Wang, X., Song, J., Sun, L., Song, M.: Amalgamating knowledge towards comprehensive classification. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2019)
26. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 3145–3153 (2017)
27. Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A.: Not just a black box: Learning important features through propagating activation differences. arXiv preprint arXiv:1605.01713 (2016)
28. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
30. Smilkov, D., Thorat, N., Kim, B., Viégas, F., Wattenberg, M.: Smoothgrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825 (2017)
31. Song, J., Chen, Y., Wang, X., Shen, C., Song, M.: Deep model transferability from attribution maps. In: Advances in Neural Information Processing Systems 32 (2019)
32. Song, J., Chen, Y., Ye, J., Wang, X., Shen, C., Mao, F., Song, M.: Depara: Deep attribution graph for deep knowledge transferability. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
33. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
34. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 3319–3328 (2017)
35. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition (June 2016)
36. Wang, Y., Xu, C., Xu, C., Tao, D.: Adversarial learning of portable student networks. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

37. Yang, Y., Qiu, J., Song, M., Tao, D., Wang, X.: Distilling knowledge from graph convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
38. Ye, J., Ji, Y., Wang, X., Gao, X., Song, M.: Data-free knowledge amalgamation via group-stack dual-gan. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
39. Ye, J., Ji, Y., Wang, X., Ou, K., Tao, D., Song, M.: Student becoming the master: Knowledge amalgamation for joint scene parsing, depth estimation, and more. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
40. Yu, X., Liu, T., Wang, X., Tao, D.: On compressing deep models by low rank and sparse decomposition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
41. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833 (2014)
42. Zhang, Q., Cao, R., Shi, F., Wu, Y.N., Zhu, S.C.: Interpreting cnn knowledge via an explanatory graph. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)