

A Learning Paradigm for Interpretable Gradients

First Author Name¹^a, Second Author Name¹^b and Third Author Name²^c

¹*Institute of Problem Solving, XYZ University, My Street, MyTown, MyCountry*

²*Department of Computing, Main University, MySecondTown, MyCountry*
{f_author, s_author}@ips.xyz.edu, t_author@dc.mu.edu

Keywords: Gradient, Class Activation Maps, Interpretability

Abstract: This paper studies deep neural networks interpretability and more specifically the production of saliency maps to explain a Convolutional Neural Network (CNN) decision. Most of the existing approaches derive from Class Activation Maps (CAM) and GradCAM. These methods combine information from fully connected layers and gradient through variations of backpropagation. We present a novel training approach to improve the gradient of a CNN in terms of interpretability. In particular, modifications to the architecture and optimization loss are presented. The gradient obtained from backpropagation at the input image level is optimized to be similar to the gradient coming from guided backpropagation. The resulting gradient is smoother and offer more interpretable power to the CNN, when applying GradCAM and some variants.

1 INTRODUCTION

2 Introduction

The key contributions of this learning interpretable gradients approach are summarized as follows:

- We introduce


3 Related Work


Interpretability or explainability of DNN decisions is a topic that receives increasing interest. As interpretability is not a simple notion to define, we recommend the work of Lipton (?) that propose some common ground, definitions and categorization for interpretability methods. For instance, transparency aims at making models simple so it is humanly possible to provide an explanation of its inner mechanism. However, post hoc method consider models as black boxes and study the activations leading to a specific output. LIME (?) and SHAP () are probably the most popular post hoc methods that are model agnostic and provide local information. Concerning CNN addressing image recognition tasks, numerous work generate


saliency maps highlighting the areas of an image that are responsible for a specific decision. Many of these methods are either based on Backpropagation and its variant or on Class Activation Maps that weight the importance of activation maps.

Restructuring in order with Topics: **Study on Activations maps** This approach towards interpreting models, considers interpretations as activations of a given network, on the input space for a model. Some of the earlier works (?, ?) laid the basis upon understanding what deeper layers on a model have learn, while building class-specific saliency maps by gradient ascent on the input space. **Gradient based approaches Latent Space based approaches**

CAM-based approaches Class Activation Maps(?) is visualization tool devised by Zhou *et al.* that utilizes linear combinations of the feature maps from the CNN, to highlight relevant regions to a given category. This approach set a line of work for the upcoming interpretability approaches, where different variants of CAM are proposed by altering the weighting coefficient for a given activation map. The direct continuation in this line of work is Grad-CAM(?), which makes use of the gradient flow produced by a given class, flowing through the final convolutional layer. It is also possible to extend this principle to multiple layers(?) and to improve sensitivity(?) and conservation(?) by the addition of axioms towards the computation(?). Moreover, Grad-CAM++(?) presented an opportu-

^a <https://orcid.org/0000-0000-0000-0000>

^b <https://orcid.org/0000-0000-0000-0000>

^c <https://orcid.org/0000-0000-0000-0000>

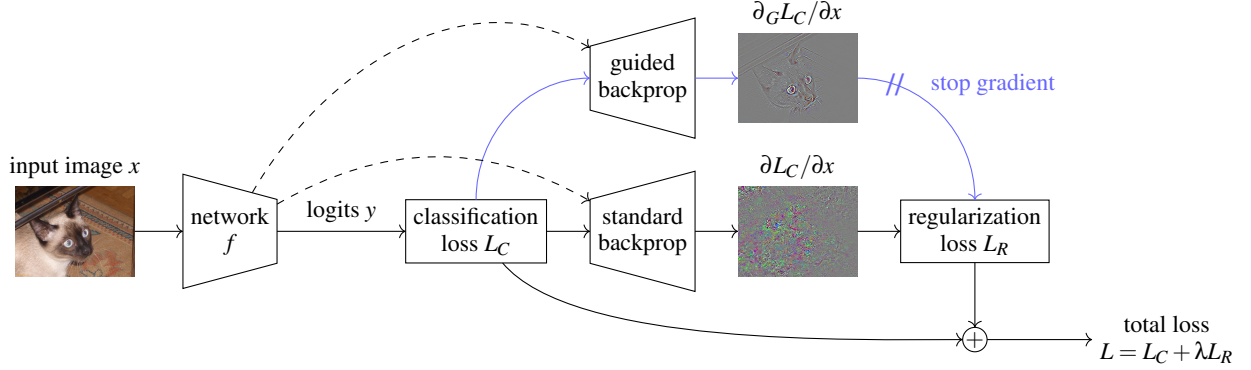


Figure 1: Pipeline of the proposed interpretable gradient learning. Given an input image x , we obtain the logit vector $y = f(x; \theta)$ by a forward pass through the network f with parameters θ and we compute the classification loss L_C by softmax and cross-entropy (7),(8). We then obtain the standard gradient $\partial L_C / \partial x$ and the guided gradient $\partial_G L_C / \partial x$ by two backward passes and, after stopping the gradient of the latter, we compute the regularization loss L_R as the error between the two (9),(??)-(14). We thus learn to denoise the standard gradient according to the guided one. Finally, the total loss is $L = L_C + \lambda L_R$ (10). Backpropagation through f is depicted “unrolled” as a decoder; there are implicit connections between the forward and backward passes (dashed lines). Learning is based on $\partial L / \partial \theta$, which involves differentiation of the entire computational graph except the guided backpropagation branch (in blue).

nity to improve the capabilities of object localization leveraging positive partial derivatives, while measuring interpretability by the introduction of metrics towards recognition and localization. It is possible to improve the gradient by adding noise(?) as proposed in (?). Most recently, Score-CAM(?) provides a gradient-free of obtaining weighting coefficients for the set of activations, by making use of the intuition proposed in Grad-CAM++. Some improvements can be produced for this approach by the addition of integration to compute scores(?) and reapplying the denoising intuition to produce sharper saliency maps(?)

3.1 Double backpropagation

Double backpropagation is a general regularization method first introduced by Drucker and Le Cun (?) to improve generalization and is later used to avoid overfitting (?), help transfer (?), cope with noisy labels (?), and recently to increase adversarial robustness (??; ??; ??; ?). This regularization aims at penalizing the $L1$ (?), $L2$, or L^∞ norm of the gradient with respect to the input image.

Our contribution measure some norm of the difference between standard and guided gradient instead.

4 BACKGROUND

4.1 Guided backpropagation (?)

The derivative of the ReLU unit $v = \text{ReLU}(u) = [u]_+ = \max(u, 0)$ with respect to its input u is $dv/du = \mathbb{1}_{u>0}$. By the chain rule, a signal $\delta v = \partial L / \partial v$ is then propagated backwards through ReLU to $\delta u = \partial L / \partial u$ as $\delta u = \mathbb{1}_{u>0} \delta v$, where the partial derivative of any scalar quantity of interest is meant, *e.g.* a loss L .

Guided backpropagation (?) changes this to $\delta_G u = \mathbb{1}_{u>0} \mathbb{1}_{\delta v>0} \delta v = \mathbb{1}_{u>0} [\delta v]_+$, essentially masking out values corresponding to negative entries of both the forward (u) and the backward (δv) signals and thus preventing backward flow of negative gradients.

Considering the partial derivative $\partial L / \partial u$ of some loss L with respect to a variable u , *e.g.* through an entire network or part of it, standard backpropagation with this particular change for ReLU units is called *guided backpropagation*. We denote the corresponding guided “partial derivative” or *guided gradient* by $\partial_G L / \partial u$. This backpropagation method allows sharp visualization of high level activations conditioned on input images.

4.2 CAM-based methods

Given a target class c and a set of 2D feature maps $\{A^k\}_{k=1}^K$ from a convolutional layer of a network, a *class activation map* (CAM) (?) is a 2D saliency map obtained in general by a linear combination of the fea-

ture maps, followed by ReLU

$$L^c = \text{ReLU} \left(\sum_{k=1}^K \alpha_k^c A^k \right). \quad (1)$$

The weight α_k^c determines the contribution of channel k to class c . This map is nonnegative, while the feature maps A^k are also assumed nonnegative as obtained by ReLU activation functions. Different CAM-based methods differ primarily in the definition of weights α_k^c .

CAM (?) originally defines α_k^c as the weight connecting channel k to class c in the classifier, assuming $\{A^k\}$ are the feature maps of the last convolutional layer, which is followed by *global average pooling* (GAP) and a linear classifier.

Grad-CAM (?) is a generalization of CAM for arbitrary networks. If y^c is the logit of class c , the weights are obtained by GAP of the partial derivatives of y^c with respect to the feature map A^k at any layer:

$$\alpha_k^c = \frac{1}{Z} \sum_{i,j} \frac{\partial y^c}{\partial A_{ij}^k}, \quad (2)$$

where A_{ij}^k denotes the value at spatial location (i, j) of feature map A^k and Z is the number of spatial locations.

Guided Grad-CAM elementwise-multiplies the saliency maps obtained by Grad-CAM and guided backpropagation, after adjusting spatial resolutions. The resulting visualizations are both class-discriminative (by Grad-CAM) and contain fine-grained detail (by guided backpropagation).

Grad-CAM++ (?) is a generalization of Grad-CAM, where partial derivatives of y^c with respect to A^k are followed by ReLU as in guided backpropagation (?) and GAP is replaced by a weighted average:

$$a_k^c = \sum_{i,j} w_{ij}^{kc} \text{ReLU} \left(\frac{\partial y^c}{\partial A_{ij}^k} \right). \quad (3)$$

The weights w_{ij}^{kc} in this linear combination are derived as

$$w_{ij}^{kc} = \frac{\frac{\partial^2 y^c}{\partial (A_{ij}^k)^2}}{2 \frac{\partial^2 y^c}{\partial (A_{ij}^k)^2} + \sum_{a,b} A_{ab}^k \frac{\partial^3 y^c}{\partial (A_{ij}^k)^3}}. \quad (4)$$

Smooth Grad-CAM++ (?) SmoothGrad (?) argues that non-linearities like ReLU activations make derivatives noisy. To improve smoothness of any

derivative $D(x)$ with respect to input image x , it takes m Gaussian noise images $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ and averages the derivatives with respect to $x + \epsilon_i$:

$$\hat{D}(x) = \frac{1}{m} \sum_{i=1}^m D(x + \epsilon_i). \quad (5)$$

Smooth Grad-CAM++ (?) applies this idea to Grad-CAM++ (?), essentially replacing all partial derivatives in (3),(4) by smoothed versions according to (5).

Score-CAM (?) Following the *increase in confidence* metric proposed in Grad-CAM++, Score-CAM defines the weights a_k^c as the increase in confidence for class c obtained by masking (element-wise multiplying) the input image x with feature map A^k :

$$a_k^c = f(x \circ s(\text{Up}(A^k)))_c - f(x_b)_c, \quad (6)$$

where Up is upsampling to the spatial resolution of x , s is linear normalization to range $[0, 1]$, \circ is the Hadamard product, f is the classifier mapping of input image to class logits and x_b is a baseline image.

Score-CAM does not require gradients to compute saliency maps, but (6) requires one forward pass through the network f for each channel k .

5 PROPOSED METHOD

Preliminaries We consider a classification network f with parameters θ , which maps an input image x to a vector of class logits $y = f(x; \theta)$. At inference, we predict the class label of maximum confidence $\arg \max_j y_j$, where y_j is the logit of class i . At training, given training images $X = \{x_i\}_{i=1}^n$ and target labels $T = \{t_i\}_{i=1}^n$, we compute the *classification loss*

$$L_C(X, \theta, T) = \frac{1}{n} \sum_{i=1}^n \text{CE}(f(x_i; \theta), t_i), \quad (7)$$

where CE is softmax followed by cross-entropy:

$$\text{CE}(y, t) = -\log \frac{e^{y_t}}{\sum_i e^{y_i}} = -y_t + \sum_i e^{y_i}. \quad (8)$$

Updates of parameters θ are then performed by an optimizer, based on the standard partial derivative (gradient) $\partial L_C / \partial \theta$ of the classification loss L_C with respect to θ , which is obtained by standard backpropagation.

However, due to nonlinearities like ReLU activations and downsampling *e.g.* by max-pooling or convolution stride > 1 , the standard gradient is noisy (?). This is mostly evident by taking and visualizing the gradient $\partial L_C / \partial x$ with respect to an input image x . By contrast, the guided gradient $\partial_G L_C / \partial x$ (?) is virtually noise-free, while preserving sharp detail. The difference of the two gradients is illustrated in 1.

Regularization The main idea of this work is to introduce a regularization loss at training, which will make the standard gradient $\partial L_C / \partial x$ behave similarly to the corresponding guided gradient $\partial_G L_C / \partial x$, while maintaining the predictive power of the classifier f . We hypothesize that, if possible, this will improve the quality of all gradients with respect to intermediate activations and therefore the quality of saliency maps obtained by CAM-based methods (??; ??; ??) and the interpretability of network f . The effect may be similar to that of SmoothGrad (??), but without the need for several forward passes at inference.

In particular, given an input image x , we perform a forward pass through f and compute the logit vectors $y_i = f(x_i, \theta)$ and the classification loss $L_C(X, \theta, T)$ (7). We then obtain the standard gradients $\delta x_i = \partial L_C / \partial x_i$ and the guided gradients $\delta_G x_i = \partial_G L_C / \partial x_i$ with respect to the input images x_i by two separate backward passes. Since the whole process will be differentiated (w.r.t. θ) at training, we stop the gradient computation of the latter, so that it only serves as a “teacher”. We define the *regularization loss*

$$L_R(X, \theta, T) = \frac{1}{n} \sum_{i=1}^n E(\delta x_i, \delta_G x_i), \quad (9)$$

where E is an error function between the two gradient images, considered below.

Algorithm 1: Interpretable gradient loss

Input: network f , parameters θ
Input: input images $X = \{x_i\}_{i=1}^n$
Input: target labels $T = \{t_i\}_{i=1}^n$
Output: loss L
 $L_C \leftarrow \frac{1}{n} \sum_i \text{CE}(f(x_i; \theta), t_i)$ ▷ class. loss (7)
foreach $i \in \{1, \dots, n\}$ **do**
 $\delta_G x_i \leftarrow (\partial_G L_C / \partial x_i)$ ▷ guided grad
 $\delta x_i \leftarrow \partial L_C / \partial x_i$ ▷ standard grad
 $L_R \leftarrow \frac{1}{n} \sum_i E(\delta x_i, \delta_G x_i)$ ▷ reg. loss (9)
 $L \leftarrow L_C + \lambda L_R$ ▷ total loss (10)

Algorithm Finally, the total loss is defined as

$$L(x, \theta, t) = L_C(x, \theta, t) + \lambda L_R(x, \theta, t), \quad (10)$$

where λ is a hyperparameter determining the regularization strength. It should be as high as possible, in the sense of not causing an increase of the classification loss and hurting training process or the final classifier accuracy. Updates of network parameters θ are now based on the gradient $\partial L / \partial \theta$ of the total loss with respect to θ , using any optimizer. At inference, one may use any method to obtain a saliency map for gradient visualization at any layer.

Our method is summarized in 1, where DETACH stops gradient computation, and illustrated in 1. Observe that the entire computational graph depicted in 1 involves one forward and two backward passes. This graph is then differentiated again to compute $\partial L / \partial \theta$, which involves one more forward and one backward pass, since the guided backpropagation branch is excluded. Thus, each training iteration incurs five passes through f , whereas standard training incurs two (one forward and one back). This amounts to $2.5 \times$ time and space overhead.

Error function Given two gradient images δ, δ' consisting of p pixels each, we consider the following error functions E in the regularization loss (9).

1. Sigmoid Masking

$$E_\sigma(\delta, \delta') = - \sum_{i=0}^p \delta' \odot \frac{e^\delta}{e^\delta + 1}. \quad (11)$$

2. Centered Sigmoid Masking

$$E_\sigma(\delta, \delta') = - \sum_{i=0}^p \delta' \odot \frac{e^{\delta - \mu_\delta}}{e^{\delta - \mu_\delta} + 1}. \quad (12)$$

We also consider the following two similarity functions, with a negative sign.

3. Histogram intersection:

$$E_{\text{HI}}(\delta, \delta') = - \sum_{i=0}^p \frac{\min(|\delta_i|, |\delta'_i|)}{\|\delta\|_1 \|\delta'\|_1}. \quad (13)$$

4. Cosine similarity:

$$E_{\text{cos}}(\delta, \delta') = - \frac{\langle \delta, \delta' \rangle}{\|\delta\|_2 \|\delta'\|_2}, \quad (14)$$

where $\langle \cdot, \cdot \rangle$ denotes inner product.

6 EXPERIMENTS

6.1 Experimental Set-up

6.1.1 Datasets

6.1.2 Settings

The evaluation of interpretability capabilities of our models is performed in terms of **Faithfulness** and **Localization**.

6.2 Faithfulness Evaluation via Image Recognition

Faithfulness evaluation as proposed in (?) offers insight about the regions of an image that are considered important for recognition, as highlighted by the saliency map L^c . Specifically, an image I is considered to produce the explanation map of class c :

$$E^c = S^c \circ I$$

Both image and explanation map are forwarded to the network to obtain the prediction scores Y_i^c and O_i^c respectively, to compute several metrics:

- **Average Drop** Aims at quantifying how much predictive power is lost when we only take into consideration the important regions of the image towards recognition. Lower is better.

$$AD(\%) = \sum_{i=1}^N \frac{\max(0, Y_i^c - O_i^c)}{Y_i^c} \quad (15)$$

- **Average Increase** Unlike Average Drop, Increase of Confidence measures the percentage of instances of the dataset where the explanation map offers a higher score than the original images for a specific class. Higher is better.

$$AI(\%) = \frac{1}{N} \sum_i^N \mathbb{1}(Y_i^c < O_i^c) * 100 \quad (16)$$

- **Average Gain**

$$AG(\%) = \sum_{i=1}^N \frac{\max(0, O_i^c - Y_i^c)}{Y_i^c} \quad (17)$$

6.3 Implementation details

6.3.1 ImageNet Experiments

(?) Off the shelf experiments, V100 32gb, batch size 100, Adam optimizer, learning rate 1e-5, 5 epochs per experiment, plateau. (Verbose)

6.3.2 CIFAR Experiments

(Verbose)