Generating Attribution Maps with Disentangled Masked Backpropagation

Adria Ruiz Antonio Agudo Francesc Moreno-Noguer
Institut de Robotica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
{aruiz,aagudo,fmoreno}@iri.upc.edu

Abstract

Attribution map visualization has arisen as one of the most effective techniques to understand the underlying inference process of Convolutional Neural Networks. In this task, the goal is to compute an score for each image pixel related to its contribution to the network output. In this paper, we introduce Disentangled Masked Backpropagation (DMBP), a novel gradient-based method that leverages on the piecewise linear nature of ReLU networks to decompose the model function into different linear mappings. This decomposition aims to disentangle the attribution maps into positive, negative and nuisance factors by learning a set of variables masking the contribution of each filter during back-propagation. A thorough evaluation over standard architectures (ResNet50 and VGG16) and benchmark datasets (PASCAL VOC and ImageNet) demonstrates that DMBP generates more visually interpretable attribution maps than previous approaches. Additionally, we quantitatively show that the maps produced by our method are more consistent with the true contribution of each pixel to the final network output.

1. Introduction

Convolutional Neural Networks (CNNs) are ubiquitous in current state-of-the-art approaches for automatic visual understanding. Despite their outstanding performance in multiple tasks [10, 14, 40], they are still characterized as black-boxes whose internal inference rules are difficult to interpret. As a consequence, the trustability of this type of models is limited and it holds back their broader adoption in applications such as autonomous driving [17] or medical diagnosis [6], where it is crucial to ensure that model decisions are reliable and not based on data artifacts or biases.

In this context, several strategies have been explored to visualize the underlying rules guiding the model's decision process [39]. Attribution map generation is one of the most effective methods for this purpose [3, 21, 25, 31, 33, 37]. This task aims to assign an score to each individual input (*i.e.*, pixels) determining their contribution to the final network output (*e.g.*, the probability for a given class). By visualizing attribution maps, it is then easy to verify whether network inference is guided by intuitive rules such as the identification of discriminative image regions related to high-level semantic concepts (see Fig. 1).

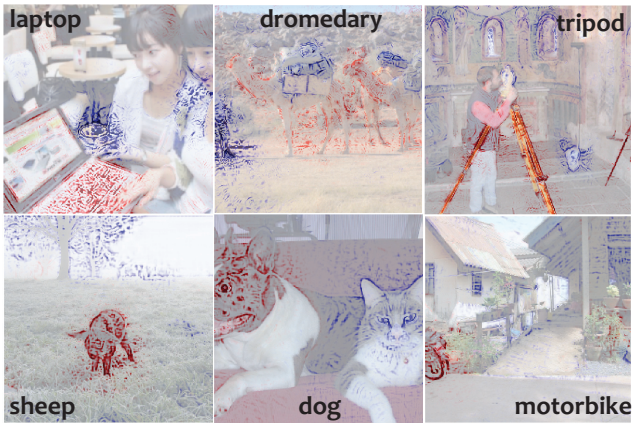

Figure 1. **Illustration of our proposed method for attribution map generation.** Given an input image, we want to obtain a score for each pixel estimating their contribution to the CNN output for a target label (in bold). Our approach is able to identify discriminative image pixels that contribute positively (red) or negatively (blue) to the prediction, and pixels corresponding to nuisance factors that have no effect on the output (white). For instance, in bottom-middle, positive attributions are assigned to the dog whereas negative scores correspond to pixels belonging to the cat. Additionally, no attributions are assigned to the non-discriminative background pixels. The disentanglement of this components produces fine-grained pixel-level attributions revealing the patterns used by the network during inference. We show that the generated attribution maps are more informative and visually interpretable than the ones obtained by previous methods.

A promising approach to generate reliable attribution maps are gradient-based techniques [28, 33, 37, 31]. To determine the importance of each pixel, these methods use different mechanisms to backpropagate the information from the output to the input image through the intermediate layers. An appealing property of gradient-based methods is that, compared to other approaches producing coarse and less informative attribution maps [21, 25], they can identify

high-frequency patterns such as edges or textures. It has been shown that this information can be relevant to fully understand the network inference process [11].

In this paper, we introduce Disentangled Masked Backpropagation (DMBP). Similar to previous gradient-based approaches, our method uses backpropagation to determine the contribution of each input pixel to the network output. However, DMBP addresses this task from a novel perspective. In particular, we use the fact that standard CNNs with ReLU non-linearities can be interpreted as piecewise linear functions where the input space is separated into different linear regions depending on the input [35]. Using this observation, DMBP decomposes output's computation into different linear mappings that are used to disentangle nuisance, positive and negative factors from the attribution map. Whereas nuisance components refer to information that have no effect on the network output, the latter factors identify the discriminative pixels providing negative or positive evidences for the target label (see again Fig. 1). The different linear mappings are identified by decomposing the network gradient into different sub-components, which are identified by learning a set of variables masking network filters during backpropagation (see Fig. 2 for an overview).

In our experiments, we validate the effectiveness of DMBP by providing qualitative and quantitative results over standard network architectures (ResNet50 and VGG16) and benchmark datasets (ImageNet and PASCAL VOC). The results demonstrate that, compared to previous methods, attribution maps produced by DMBP are more consistent with the true contribution of each pixel to the network output. Moreover, we show that our results are more informative and visually interpretable.

2. Related Work

Attribution Map Generation is one of the most effective strategies to understand the inference process of a CNN for a given input. For this purpose, perturbation-based methods measure the contribution of input pixels by observing the effect of excluding or including them during inference. These approaches use different mechanisms to generate binary masks defining image regions that are perturbed for network evaluation. Prediction Difference Analysis [42] and Occlusion [37] use a sliding window approach to set to zero image patches and measure the effect on the CNN output. RISE [21] generates random binary masks and average them according to the target class probability. LIME [23] and KernelShap [19] weights image super-pixels according to a surrogate model that estimates the effect on the CNN output when they are removed. In contrast to these brute-force strategies, Meaningful [9] and Extremal Perturbations [8] pose this task as a learning problem where the mask is optimized to minimize the target label probability. However, while the previous methods have shown promising results, the generated attribution maps are sensitive to different hyper-parameters [5, 2] controlling factors such as: (i) the type of image perturbation [9], (ii) the extracted super-pixels [19, 23] (iii) the sampling process over the masks [42, 21, 23] or (iiii) sparsity and smoothing constraints [8, 9]. These parameters can be difficult to validate in practice given the absence of an objective ground-truth.

Another popular approach to generate attribution maps is by exploiting the information contained in the intermediate network layers. In particular, Class Activation Maps [41] uses the weights of the final classifier to compute a linear combination of the feature maps in the last average pooling layer. GradCam [25] considered a similar approach with a linear combination determined by the gradients of the output w.r.t. the last feature map. Score-CAM [34] uses the intermediate layer activations to generate attribution maps following a similar strategy than perturbation-based methods. Full-gradient [32] uses the gradient of the bias terms w.r.t. the output in order to generate attributions. More recently, Principal Feature Visualization [4] visualized the information of the last CNN layer through a PCA over the corresponding feature map. Finally, [22] combines multiple attribution maps generated from the gradient information of the output w.r.t. the intermediate layer parameters. Despite these approaches typically involve less hyper-parameters than perturbation-based methods, the information of intermediate layers is visualized by up-sampling it to the resolution of the original input image. As a consequence, the generated attribution maps are coarse-grained and do not reveal cues such as texture or edges that can be critical to understand the network inference process [11].

Gradient based methods for generating attribution maps are motivated by the fact that the gradient of a CNN output w.r.t. the input image is related with the contribution of each pixel to the final prediction. Based on this observation, [28] proposed to directly use the network gradient to compute the importance of each pixel. However, the results of this approach tend to be too noisy to be easily interpreted. To overcome this limitation, several approaches average multiple gradients computed w.r.t. a set of modified input images. In particular, Integrated Gradients [33] considers a set of interpolations between the original and a reference input (*e.g.*, a zero image). XRAI [16] applies this framework to assign an score to different super-pixels. BIG [12] uses a set of blurred images as reference inputs. Finally, SmoothGrad [30] averages multiple gradients resulting from evaluating different inputs corrupted with Gaussian noise.

Rather than averaging multiple gradients, other approaches attempt to filter the non-relevant information during backpropagation by modifying the activation function derivatives [2]. For instance, DeconvNet [37] applies a ReLU activation to the gradient of each intermediate layer. Guided Backpropagation [31] follows a similar strategy but

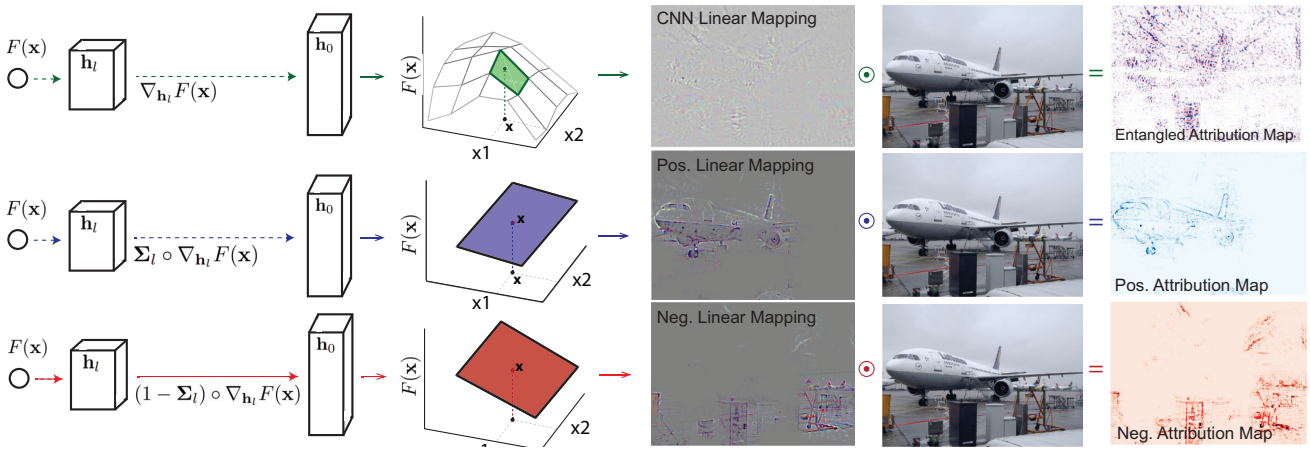

Figure 2. **Overview of Disentangled Masked Backpropagation for generating attribution maps. Top:** Given a function $F(\mathbf{x})$ modelled by a CNN with ReLU non-linearities, the network output for a given image can be computed by applying a linear mapping over the input. This mapping is equivalent to the output gradients w.r.t. the input. Then, the attribution map indicating the contribution of each pixel can be computed as an element-wise multiplication of the mapping and the image pixels. As can be seen, however, this strategy typically produces noisy results that are difficult to visually interpret **Middle and Bottom:** DMBP learns a set of variables weighting the contribution of each network filter during backpropagation. The optimization of these variables is guided by a loss which decomposes the original function into different linear mappings disentangling positive and negative attributions and removing nuisance factors from the attribution maps.

the derivative is also zeroed if the input to the corresponding layer is negative. Guided GradCam [25] combines Guided Backpropagation attributions with the coarse-grained maps of GradCam. Finally, methods such as Excitation Backpropagation [38], Layer-wise Relevance Propagation [3], DeepLift [26], DeepShap [19], Deep Taylor Decomposition [20] or PatternAttribution [18] employ different gradient computation rules to propagate attributions across layers.

The proposed Disentangled Masked Backpropagation draws inspiration on the methods that use a gradient-based strategy to generate attribution maps. A fundamental difference, however, is that DMBP does not rely on hand-crafted backpropagation rules as in [38, 3, 37, 31]. Instead, it optimizes a set of variables masking the individual network filters during gradient computation. While learning function derivatives has been recently explored in LPR [36], this method uses the modified gradients to generate a binary-mask similarly to perturbation-based methods. DMBP, instead, is the first approach to optimize backpropagation rules to explicitly decompose the network function into a set of linear mappings disentangling positive, negative and nuisance factors from the attribution maps.

3. Disentangled Masked Backpropagation

In the following, we introduce the formal definition of attribution map used in our framework. Let us consider a linear model of the form $y = \mathbf{c}^T\mathbf{x}$, where $y \in \mathbb{R}$ is the output (*e.g.*, an score for a given class) and $\mathbf{c} \in \mathbb{R}^{d_0}$ is a linear mapping applied to the input $\mathbf{x} \in \mathbb{R}^{d_0}$ (*e.g.*, a vectorized RGB image). Given any input-output pair $\{\mathbf{x}, y\}$, we can compute an attribution map $\mathbf{a} \in \mathbb{R}^{d_0}$ as $\mathbf{a}(\mathbf{x}) = \mathbf{c} \odot \mathbf{x}$, where $\odot$ is the Hadamard product and $y = \sum_i \mathbf{a}_i$. As $\mathbf{x}$ is an image, $\mathbf{a}$ can be visualized in order to identify the contribution of each input pixel to the output. From now on, we use the previous definition of attribution map for DMBP and the rest of gradient-based approaches.

In this section, we show that standard ReLU networks model a linear function for each input $\mathbf{x}$ (Section 3.1). For the sake of simplicity, we start considering networks with fully-connected layers and without bias terms. In Section 3.2, we explain how DMBP uses this input-dependent linearization in order to disentangle positive, negative and nuisance factors from the attribution maps. Finally, in Sections 3.3 and 3.4, we generalize our framework to the case of networks with bias terms and CNNs, respectively.

3.1. Linearizing ReLU Neural Networks

Let us consider a neural network with L fully-connected layers defining a function $F : \mathbb{R}^{d_0} \rightarrow \mathbb{R}$ as:

$$y = F(\mathbf{x}) = \mathbf{w}^T \big[f_L \circ \ldots \circ f_l \circ \ldots \circ f_2 \circ f_1(\mathbf{x}) \big], \quad (1)$$

where $\mathbf{w} \in \mathbb{R}^{d_L}$ represents a final filter computing an output from the last hidden layer (*e.g.*, the target label score before applying a softmax). Additionally, each intermediate layer f is defined as the composition of a linear function and a ReLU non-linearity as:

$$\mathbf{h}_l = f_l(\mathbf{h}_{l-1}) = \phi(\mathbf{W}_l \mathbf{h}_{l-1}), \quad (2)$$

where $\mathbf{h}_l \in \mathbb{R}^{d_l}$ are the intermediate layer activations, $\mathbf{W} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\phi(\cdot) = \max(\cdot, 0)$. Given previous definitions,

we can express Eq. (2) as:

$$\mathbf{h}_l = \hat{\mathbf{W}}_l \mathbf{h}_{l-1} = \text{diag}(\mathcal{H}(\mathbf{W}_l \mathbf{h}_{l-1}))\mathbf{W}_l \mathbf{h}_{l-1}, \quad (3)$$

where $\mathcal{H}(\cdot)$ denotes the Heaviside step function applied to all the elements in a vector. More intuitively, we model the ReLU operation as a diagonal binary matrix masking the subset of filters in $\mathbf{W}_l$ that produces negative elements via $\mathbf{W}_l \mathbf{h}_{l-1}$. As a result, we can express the composition of the linear mapping and the ReLU as a single matrix $\hat{\mathbf{W}}_l$.

From Eqs. (1) and (3), it is easy to see that the network output is computed by applying a composition of linear transformations $\hat{\mathbf{W}}_l$ over the input $\mathbf{x}$ as:

$$y = \mathbf{w}^T \Big[\hat{\mathbf{W}}_L \dots \hat{\mathbf{W}}_2 \hat{\mathbf{W}}_1\Big]\mathbf{x} = \mathbf{w}^T \mathbf{H}_L \mathbf{x}. \quad (4)$$

where $\mathbf{H}_L \in \mathbb{R}^{d_L \times d_0}$. Note that for any linear function of the form $F(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, we have $\mathbf{c} = \nabla_{\mathbf{x}} F(\mathbf{x})$. Therefore, the vector $\mathbf{w}^T\mathbf{H}_L \in \mathbb{R}^{d_0}$ in Eq. (4) is equivalent to the gradient of the network's output w.r.t. the input. An attribution map $\mathbf{a}(\mathbf{x}) \in \mathbb{R}^{d_0}$ for a given image can thus be computed as:

$$\mathbf{a}(\mathbf{x}) = [\mathbf{w}^T \mathbf{H}_L] \odot \mathbf{x} = \nabla_{\mathbf{x}} F(\mathbf{x}) \odot \mathbf{x}. \quad (5)$$

3.2. Attribution Map Disentanglement

Motivation. Using the output gradient to compute attribution maps was initially proposed in [28]. However, this strategy typically produces noisy results that are difficult to visually interpret (see Fig. 2-Top). To understand this phenomena, we need to analyse the role of the matrix $\mathbf{H}_L$ in Eq. (4). In particular, it can be interpreted as a linear mapping computing the last layer features $\mathbf{h}_L$ from the input $\mathbf{x}$. However, note that the resulting features entangle both discriminative and non-relevant information encoded by the network during inference. Nuisance components are thus also visualized in the attribution map, masking the discriminative factors that truly contribute to the model's output.

Motivated by this observation, DMBP decomposes Eq. (4) into three different terms:

$$\mathbf{w}^T \mathbf{H}_L \mathbf{x} = \mathbf{w}^T \big(\mathbf{H}_L^+ + \mathbf{H}_L^- + \mathbf{H}_L^\sim\big)\mathbf{x}, \quad (6)$$

where $\mathbf{H}_L^+$ and $\mathbf{H}_L^-$ are linear mappings producing features that contribute positively and negatively to the output, respectively. In contrast, $\mathbf{H}_L^\sim$ aims to extract the non-discriminative features.

Using this decomposition, an attribution map without nuisance factors can be computed as:

$$\mathbf{a}(\mathbf{x}) = [\mathbf{w}^T \mathbf{H}_L^+] \odot \mathbf{x} + [\mathbf{w}^T \mathbf{H}_L^-] \odot \mathbf{x}. \quad (7)$$

Filter decomposition. To obtain the decomposition in Eq. (6), we use the fact that the feature extractor $\mathbf{H}_L$ in Eq. (4) is defined as a product of matrices $\hat{\mathbf{W}}_l$. We can therefore decompose each of these linear mappings as:

$$\hat{\mathbf{W}}_l = \hat{\mathbf{W}}_l^+ + \hat{\mathbf{W}}_l^- = \mathbf{\Sigma}_l \hat{\mathbf{W}}_l + (\mathbf{I} - \mathbf{\Sigma}_l)\hat{\mathbf{W}}_l, \quad (8)$$

where $\mathbf{I}$ is the identity and $\mathbf{\Sigma}_l = \text{diag}(\boldsymbol{\sigma}_l) \in \mathbb{R}^{d_l \times d_l}$ is a diagonal matrix whose entries $\boldsymbol{\sigma}_l \in [0,1]^{d_l}$ are vectors of learnable parameters.

Denoting $\boldsymbol{\sigma} = \{\boldsymbol{\sigma}_L, \dots, \boldsymbol{\sigma}_1\}$, the network output in Eq. (4) can be explicitly decomposed into positive, negative and nuisance terms as:

$$\begin{aligned} y &= \mathbf{w}^T \big(\mathbf{H}_L^+ + \mathbf{H}_L^- + \mathbf{H}_L^\sim\big)\mathbf{x} \\ &= y^+(\boldsymbol{\sigma}) + y^-(\boldsymbol{\sigma}) + y^\sim(\boldsymbol{\sigma}) \\ &= \mathbf{w}^T \Big[\mathbf{\Sigma}_L \hat{\mathbf{W}}_L \dots \mathbf{\Sigma}_1 \hat{\mathbf{W}}_1\Big]\mathbf{x} \\ &\quad + \mathbf{w}^T \Big[(\mathbf{I} - \mathbf{\Sigma}_L)\hat{\mathbf{W}}_L \dots (\mathbf{I} - \mathbf{\Sigma}_1)\hat{\mathbf{W}}_1\Big]\mathbf{x} \\ &\quad + \mathbf{w}^T \mathbf{H}_L^\sim \mathbf{x}, \end{aligned} \quad (9)$$

where the masks $\mathbf{\Sigma}_l$ and $(\mathbf{I} - \mathbf{\Sigma}_l)$ select the set of filters for each layer producing features that have a positive or negative effect on the output, respectively. In contrast, $\mathbf{H}_L^\sim$ models the non-discriminative features.

Learning objective. In order to learn the optimal parameters $\boldsymbol{\sigma}$ for a given input image $\mathbf{x}$, DMBP optimizes:

$$\min_{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \dots, \boldsymbol{\sigma}_L} y^-(\boldsymbol{\sigma}) - y^+(\boldsymbol{\sigma}) + \|y^\sim(\boldsymbol{\sigma})\|_1, \quad (10)$$

where we aim to maximize and minimize the positive and negative terms in Eq. (9), respectively. Additionally, the term $\|y^\sim(\boldsymbol{\sigma})\|_1$ encourages nuisance factors to have a negligible effect on y. During optimization, we ensure the constraint $\boldsymbol{\sigma}_l \in [0,1]^{d_l}$ by applying a sigmoid function over the set of learned scalar parameters.

Optimization via masked backpropagation. Analogously to Eq. (4), the positive and negative terms in Eq. (10) are linear w.r.t. the input and thus, they can be expressed as:

$$y^+(\boldsymbol{\sigma}) = \nabla_{\mathbf{x}} F^+(\mathbf{x})^T \mathbf{x} \quad , \quad y^-(\boldsymbol{\sigma}) = \nabla_{\mathbf{x}} F^-(\mathbf{x})^T \mathbf{x} \quad (11)$$

where $\nabla_{\mathbf{x}} F^+(\mathbf{x})$ is obtained by performing a backward pass over the network while multiplying the filters of each layer by $\mathbf{\Sigma}_l$. Similarly, $\nabla_{\mathbf{x}} F^-(\mathbf{x})$ can be obtained with another backward pass using $(\mathbf{I} - \mathbf{\Sigma}_l)$. Finally, the nuisance term does not require any explicit computation since it can be estimated through the network output and the previous computed terms as $y^\sim(\boldsymbol{\sigma}) = y - y^+(\boldsymbol{\sigma}) - y^-(\boldsymbol{\sigma})$. Upon the definition of these computations, the parameters $\boldsymbol{\sigma}_l$ for each layer can be optimized by minimizing the loss function in Eq. (10) using standard gradient descent.

3.3. Incorporating Bias Terms

In previous sections, we have obviated the biases terms for each filter that are typically used in neural networks. To take them into account, we shall modify Eq. (3) as:

$$\mathbf{h}_l = f_l(\mathbf{h_{l-1}}) = \phi(\mathbf{W}_l\mathbf{h}_{l-1} + \mathbf{b}_l), \tag{12}$$

where $\mathbf{b}_l$ is the bias term of the filter $\mathbf{W}_l$.

Similar to Eq. (4), it is easy to show that, in this case, the network function can be linearized for a given input as:

$$y = \hat{\mathbf{w}}^T \left[\hat{\mathbf{W}}_L \dots \mathbf{W}_1\right]\mathbf{x} + \sum_{l=2}^{L} \hat{\mathbf{w}}^T \left[\hat{\mathbf{W}}_L \dots \hat{\mathbf{W}}_l\right]\mathbf{b}_{l-1} \tag{13}$$

where $\hat{\mathbf{W}}_l = \mathbf{W}_l \operatorname{diag}(\mathcal{H}(\mathbf{h}_{l-1} + \mathbf{b}_{l-1}))$, and $\hat{\mathbf{w}} = \hat{\mathbf{w}} \operatorname{diag}(\mathcal{H}(\mathbf{h}_L))$. See Appendix A for more details. Note that the output y is now obtained by applying a set of linear mappings over the input $\mathbf{x}$ and each bias term $\mathbf{b}_l$. Yet, the resulting function is again linear with respect to $\mathbf{x}$ and $\mathbf{b}_{0:L}$. Therefore, we can also express Eq. (13) using the output gradients w.r.t. the input and biases as:

$$y = \nabla_{\mathbf{x}} F(\mathbf{x})^T \mathbf{x} + \sum_{l=1}^{L} \nabla_{\mathbf{b_l}} F(\mathbf{x})^T \mathbf{b}_l. \tag{14}$$

While the previous expression was firstly developed in [32] by using a different derivation, we use Eq. (14) in order to compute the DMBP decomposition in Eq. (9) for neural networks with bias terms. Concretely, we follow the same procedure described in Sec. 3.2. However, the gradients w.r.t. the biases need to be computed during the two independent backward passes using $\mathbf{\Sigma}$ and $(\mathbf{I} - \mathbf{\Sigma})$. This is required to compute the contribution of the bias terms in Eq. (14).

3.4. DMBP for Convolutional Neural Networks

Masking Convolutional Filters. CNNs compute intermediate feature maps by applying a convolutional layer of the form $\mathbf{h_l} = \phi(\mathbf{W}_l * \mathbf{h}_{l-1})$. In this case, the composition of the convolutional operator and the ReLU can be also expressed as a single linear mapping:

$$\mathbf{h_l} = \phi(\mathbf{W}_l * \mathbf{h}_{l-1}) = \mathcal{H}(\mathbf{W}_l * \mathbf{h}_{l-1}) \odot (\mathbf{W}_l * \mathbf{h}_{l-1}). \tag{15}$$

The DMBP decomposition in Eq. (9) can be also applied to convolutional layers as follows. Firstly, the positive term $y^+(\boldsymbol{\sigma})$ can be obtained by multiplying $\mathbf{\Sigma}_l$ by the resulting feature maps after each convolutional and ReLU:

$$\mathbf{h_l} = \mathbf{\Sigma}_l \circ \mathcal{H}(\mathbf{W}_l * \mathbf{h}_{l-1}) \odot (\mathbf{W}_l * \mathbf{h}_{l-1}), \tag{16}$$

where $\mathbf{\Sigma}_l$ is a tensor of the same dimension than $\mathbf{h}_l$. Intuitively, this is equivalent to mask the applied filters $\mathbf{W}_l$ independently for each spatial position and channel of the input feature map. In this manner, the term $y^+(\boldsymbol{\sigma})$ can be also computed as in Eq. (11), where $\nabla_{\mathbf{x}} F^+(\mathbf{x})$ is obtained with a single backward pass over the network by modifying the gradients for each intermediate layer as $\mathbf{\Sigma}_l \circ \nabla_{\mathbf{h}_l} F(\mathbf{x})$. Similarly, the negative term $y^-(\boldsymbol{\sigma})$ can be computed using $(\mathbf{I} - \mathbf{\Sigma}_l)$ during backpropagation. Pseudo-code for DMBP optimization is provided in Appendix B.

Applying DMBP for other layers. Besides convolutions and ReLUs, standard CNNs also incorporate Batch Normalization (BN) [15] or residual layers [13]. Fortunately, the use of these layers does not requires any modification into our proposed framework. The reason is that they can also be modelled as linear mappings over the input and thus, the network function can still be linearized as in Eq. (14). During evaluation, BN can be fused with its previous convolution by modifying its filters and bias terms.[1]. On the other hand, residual layers of the form $\mathbf{h_l} = \phi(\mathbf{W}_l\mathbf{h}_{l-1}) + \mathbf{h}_{l-1}$ can be represented by a linear mapping $\mathbf{h_l} = (\hat{\mathbf{W}}_l + \mathbf{I})\mathbf{h}_{l-1}$, where $\hat{\mathbf{W}}_l$ is defined in Eq. (3).

4. Experiments

Datasets and models. We conduct our experiments over benchmark datasets and architectures for image classification. In particular, we use the validation sets of ImageNet [24] and VOC2012 [7]. As baseline models, we use two extensively used CNN architectures: ResNet50 [13] and VGG16 [29]. Over ImageNet, we use the pretrained models in the Torchvision library for both architectures. In VOC2012, we use the models trained in [38].

Baselines. We compare DMBP with 11 previous approaches for attribution map generation, including state-of-the-art methods. Given that DMBP is a gradient-based approach, we focus on the comparisons with previous methods following this strategy: Grad [28], Integrated Gradients (IG) [33], Smooth Gradients (SG) [30], Blurred Integrated Gradients (BIG) [12], DeepLift (DL) [27], Gradient Backpropagation (GBp) [31] and Guided GradCam (GGC) [25]. As a reference, we also compare our method with GradCam (GC) [25] and FullGradients (FG) [32], which use the information in intermediate layers to compute the attribution maps. Finally, we also provide comparisons with the perturbation-based approaches RISE [21] and LPR [36].

Implementation details and hyper-parameters. We use a PyTorch implementation for DMBP and the rest of compared methods. For BIG [2] and FG [3], we integrate the code provided by the authors. We use our own implementation of LPR given that no code is publicly available. For the rest of methods, we use the implementations in the Captum.ai and TorchRay libraries. The hyper-parameters for all the

[1] https://nenadmarkus.com/p/fusing-batchnorm-and-conv/
[2] https://github.com/PAIR-code/saliency
[3] https://github.com/idiap/fullgrad-saliency

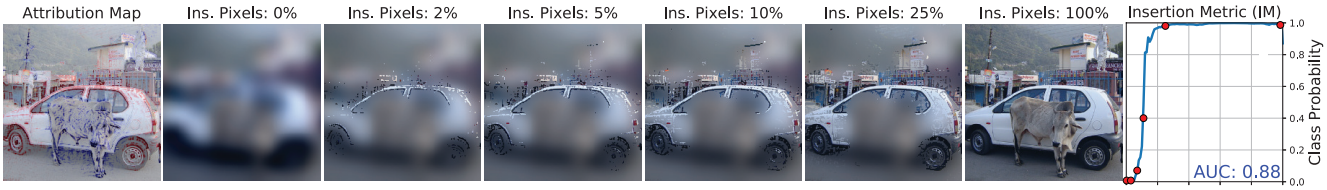

Figure 3. **Illustration of the Insertion Metric (IM) used in our experiments.** See text for details.

DB/Model	Metric	ND	DMBP^{+}	DMBP$^{+,-}$	DMBP^{All}
VOC/VGG16	IM	.36	**[.62]**	.51	**.57**
	cIM	.09	.06	**[.18]**	**.16**
VOC/RN50	IM	.30	.58	**.59**	**[.61]**
	cIM	.15	.15	**[.22]**	**[.22]**
ImNet/VGG16	IM	.24	.37	**.37**	**[.39]**
ImNet/RN50	IM	.26	.50	**.51**	**[.57]**

Table 1. **Results for DMBP variants optimized with different disentanglement losses.** See text for details. Best method is indicated with brackets. Second and best are indicated in bold.

compared methods are set to the default values suggested in the original papers. For DMBP, we use RMSProp with a learning rate of 0.01 as an optimizer to minimize the loss in Eq. (10). No weight decay is applied. A total number of 200 iterations are executed during gradient-descent. The optimization for a given 224×224 input image requires $\sim 20s$ on a NVIDIA 2080 Ti GPU. This time is higher than the needed by other gradient-based approaches. However, our primary goal is to generate accurate attribution maps rather than their efficient computation. Our code implementing DMBP is publicly available in this repository.

4.1. Evaluation Metrics

Evaluating attribution maps is challenging given the absence of objective ground-truth. Previous works have attempted to evaluate them by using object bounding-boxes provided by human annotators [25, 38] or conducting user-studies [19]. However, human-based evaluation can be flawed and misleading [1], given that the perceived discriminative pixels can differ between humans and CNNs.

Insertion Metric (IM). To overcome this limitation, we use the insertion metric proposed in [21]. This metric does not rely on human annotations and computes a score for each image as follows. Given an attribution map generated for a target label, the pixels are ordered in decreasing order according to their attribution value. Then, an increasing percentage of pixels is added iteratively to a reference blurred image and the probability of the target label is evaluated with the network. Finally, the AUC over the probabilities is computed to obtain a single score for the image (see Fig. 3 for an illustration). Intuitively, this metric attempts to measure if the pixels with large attribution values contribute positively to the network output. Different from [21] where the attribution maps were generated for the label with a highest probability, we use the ground-truth classes provided in each dataset as the target label. This results in a more challenging problem because the class can be predicted with a low probability. Additionally, using the ground-truth annotations as target labels allows us to use the following complementary metric to IM.

Complementary Insertion Metric (cIM). In IM, the first inserted pixels are the ones with highest attributions. As a consequence, this metric is not appropriate to evaluate the information provided by pixels with negative scores. We expect such pixels to correspond to regions that are discriminative but provide negative evidences for the target label. To evaluate these attributions, we use an alternative metric that can be applied to images annotated with multiple classes and which we refer to as Complementary Insertion Metric (cIM). In particular, we follow the same procedure as in IM but the pixels with lower attributions are inserted first. Then, the AUC is computed by evaluating the probabilities for all the ground-truth labels that are different from the target class used to compute the attribution map. Therefore, a high cIM indicates that pixels with negative attributions correspond to discriminative regions providing positive evidences for the complementary classes in the image. Consequently, these regions provide negative evidences for the target class. We do not compute cIM for ImageNet given that images are only labelled with a single class.

4.2. Evaluating Attribution Map Disentanglement

In this experiment, we evaluate the effect of disentangling positive, negative and nuisance factors on the generated attribution maps. For this purpose, we compare different variants of DMBP optimized with ablated versions of our loss defined in Eq. (10). Concretely, we use: (i) A loss maximizing only the positive term y^{+}. (ii) The same objective but also minimizing y^{-}. (iii) The original loss that also takes into account the nuisance term $y^{\sim}$. From now on, we refer to these approaches as DMPB^{+}, DMBP$^{+,-}$ and DMBP$^{\text{All}}$, respectively. Note that DMPB^{+} is optimized to identify only positive factors; DMBP$^{+,-}$ aims to disentangle positive and negative factors; DMBP$^{\text{All}}$ also seeks to remove nuisance factors. In addition, we also evaluate attribution maps generated by the vanilla approach in Eq. (5), where factors are not disentangled. We refer to this method as ND. For a faster experimentation, we use a subset of 5K images for ImageNet with five random images per class.

DB/Model	Metric	Grad [28]	IG [33]	SG [30]	BIG [12]	DL [27]	GBp [31]	GGC [25]	LPR [36]	RISE [21]	FG [32]	GC [25]	DMBP
VOC/VGG	IM	0.36	0.43	**0.55**	0.36	0.41	0.32	0.35	0.40	0.41	0.28	0.35	**[0.57]**
	cIM	0.09	0.06	0.06	0.08	0.07	0.06	0.12	0.07	**0.13**	0.10	**0.13**	**[0.16]**
VOC/RN50	IM	0.30	0.36	0.47	0.27	0.34	0.39	0.50	0.40	**0.51**	0.44	**0.51**	**[0.61]**
	cIM	0.15	0.16	0.15	0.15	0.15	0.13	0.19	0.14	0.21	0.14	**0.21**	**[0.22]**
INet/VGG	IM	0.23	0.28	0.38	0.21	0.30	0.32	0.36	0.28	0.28	**0.43**	**[0.48]**	0.41
INet/RN50	IM	0.26	0.30	0.40	0.27	0.29	0.43	0.50	0.33	0.33	0.53	**0.55**	**[0.56]**
	Avg	0.23	0.27	0.34	0.22	0.26	0.28	0.34	0.27	0.31	0.32	**0.37**	**[0.42]**

Table 2. **Results obtained by DMBP and other state-of-the-art methods.** Metrics are shown for all the evaluated datasets and network models. Best method is indicated with brackets. Second and best are indicated in bold.

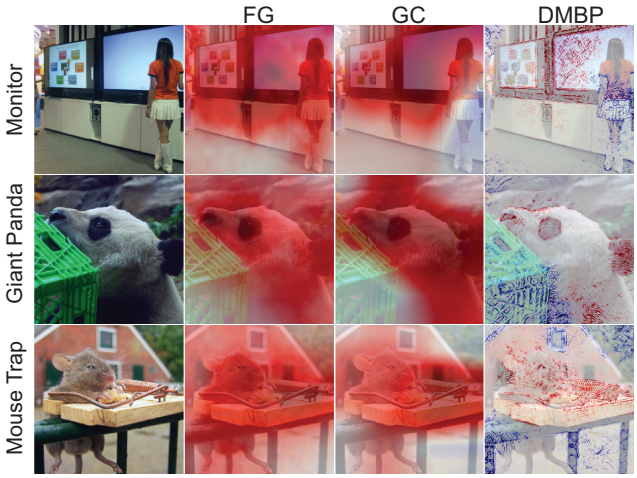

Figure 4. **Comparison between attribution maps generated by FG, GradCam and the proposed DMBP.** The latter generates fine-grained pixel attributions that are more informative and interpretable than the coarse results produced by the other methods.

Table 1 summarizes the results for the different evaluated metrics, datasets and models. As we can observe, ND yields significantly worse results compared to the different DMBP versions in most metrics. The reason is that positive, negative and nuisance factors are not disentangled in this case. This generates noisy results where attributions do not correctly identify the contribution of each pixel to the model's ouput. On the other hand, DMBP$^{+,-}$ consistently outperforms DMBP^{+} in all the cases except for the IM metric in VGG16 over VOC. This improvement is because DMBP^{+} ignores the negative factors during optimization. As a consequence, the discriminative patterns having a negative impact on the model's output are not correctly identified. This is clearly seen by observing the poor cIM values obtained by DMBP^{+}. Finally, DMBP^{All} achieves comparable or better results than DMBP$^{+,-}$ in most cases. This demonstrates the importance of removing non-discriminative factors by minimizing the effect of the nuisance term $y^{\sim}$.

4.3. Comparison with state of the art

Comparison with gradient-based methods. As can be obzerved in Table 2, DMBP consistently outperforms the rest of compared gradient-based approaches (Grad, IG, SG, BIG, DL, GBp and GGC). To provide further insights, Fig. 5 shows qualitative results obtained with DMBP and the alternative gradient-based methods with better performance. As shown, IG, SG and DL produce noisy visualization, in which positive and negative attributions are mixed. The reason is that these approaches do not explicitly disentangle the discriminative and non-relevant factors. In contrast, GBp uses hand-crafted backpropagation rules to identify image regions corresponding only to positive factors. However, this method produces attributions where pixels that do not belong to the target class are also assigned with positive attributions. Finally, GGC uses the coarse feature maps produced by GradCam to filter the attributions generated by GBp. Still, this strategy is not able to identify image pixels that have a negative contribution to the network output. In contrast to IG, SG and DL, our method produces more interpretable attribution maps by removing the non-discriminative factors. Additionally, compared to GGC and GBp, our approach correctly identifies factors that have a negative effect on the output.

Comparison with other approaches. Results in Table 2 also demonstrate the advantages of DMBP over the approaches using intermediate layer information GradCam and FG. More concretely, DMBP obtains the best average performance and the top-scoring results in all the metrics except for VGG16 over ImageNet, where FG and GC outperform our approach. Nevertheless, these two methods generate attribution maps by upsampling the information extracted from intermediate layers. As shown in Fig. 4, this results in coarser and less informative visualizations than those obtained with our method. Whereas coarse attribution maps can be potentially applied to specific downstream tasks such as weakly-supervised object localization, reliable network intepretability requires fine-grained results providing detailed information about the visual cues that the model exploits during inference. As can be observed, this is the case for DMBP visualizations, which provide fine-grained pixel-level attributions identifying high-frequency information such as object edges or textures. To conclude, our method consistently outperforms the perturbation-based methods RISE and LPR. The latter also optimizes function derivatives for backpropagation. However, the modified gradients are used to generate binary masks corrupting the original image. In contrast, DMBP achieves better performance by using the modified gradients to explicitly dis-

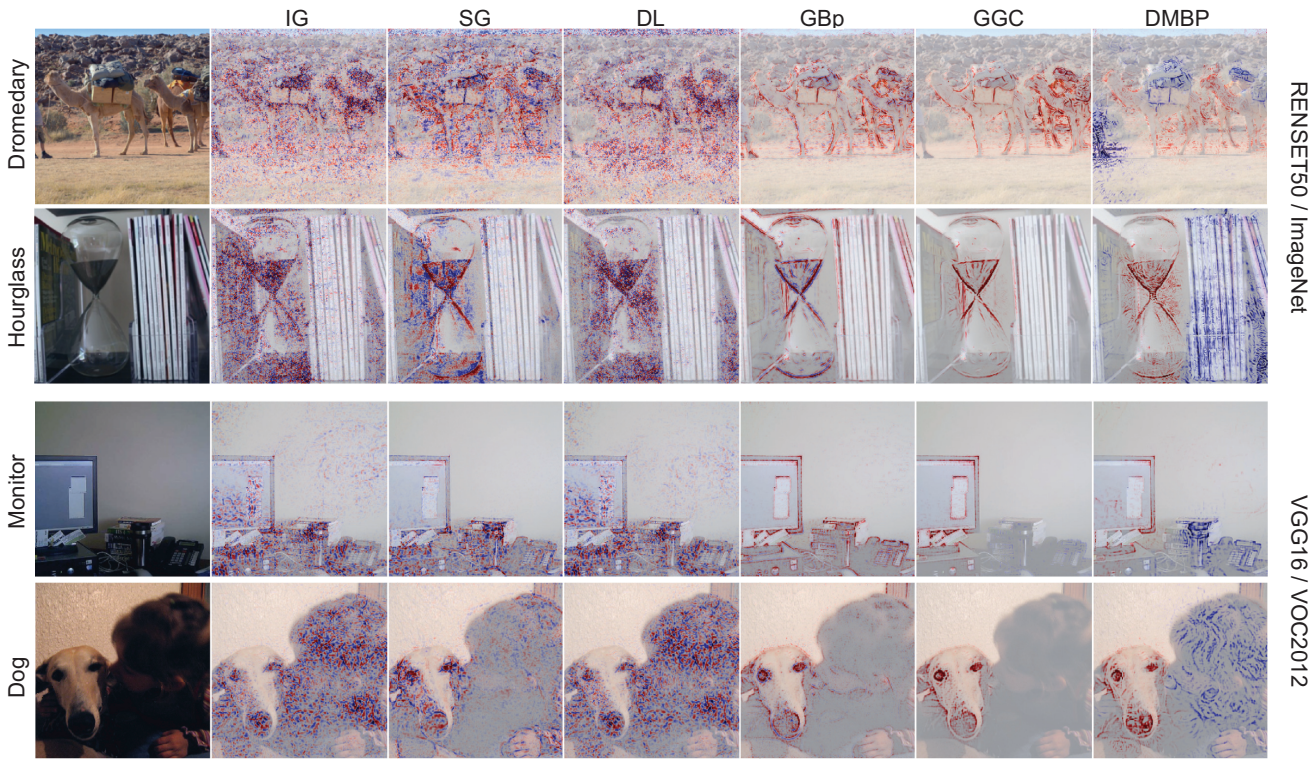

Figure 5. **Qualitative results for DMBP and alternative gradient-based approaches.** More results provided in suppl. material.

	VOC		IMNet	
	VGG16	**RN50**	**VGG16**	**RN50**
GBp [31]	0.99	0.99	0.99	0.99
GGC [25]	0.25	0.28	0.35	0.45
DMBP (Ours)	**0.02**	**-0.03**	**0.00**	**0.00**

Table 3. Correlation between the attributions obtained from the original network and for one with reinitialized parameters.

entangle positive, negative and nuisance factors.

Attributions sensitivity to layer reinitialization. DMBP generates fine-grained attributions where visual cues such as edges can be clearly identified. The identification of this intuitive patterns is also observed in gradient-based approaches such as Gradient Backpropagation and Gradient GradCam. However, [1] showed that these approaches suffer from a critical weakness: the attribution maps are not sensitive to the reinitialization of network parameters. As a consequence, they cannot be identifying the discriminative regions explaining the inference process. In order to evaluate if DMBP suffers from this limitation, we perform a sanity check proposed in [1]. In particular, we compute the rank correlation between the original attribution maps and the ones generated by a network where the last layer parameters are randomly reinitialized using a normal distribution. Table 3 shows the obtained results. As can be observed, the high correlations obtained by GBp and GGC indicate that these methods generate attribution maps that are not sensitive to model reinitialization. In contrast, DMBP obtains almost a zero correlation for all cases, showing that the identified edges are truly dependent on network parameters.

5. Conclusions

We have presented Disentangled Masked Backpropagation, a novel gradient-based approach for attribution map generation. In contrast to previous methods, DMBP leverages on the piecewise linear nature of ReLU neural networks to disentangle positive, negative and nuisance factors from the attribution maps. Our experiments demonstrate that, compared to previous state-of-the-art methods, DMBP produces fine-grained attribution maps that are more visually interpretable and identify better the contribution of each pixel to the network output. Whereas we have focused on standard CNN architectures employing ReLU activations, our framework can also be applied to networks with other types of piecewise linear activations such as the Leaky-ReLU. Last but not least, other non-linearities such as the sigmoid or the hyperbolic tangent could be also introduced by modelling them with piecewise linear approximations.

Acknowledgments. This work has been partially funded by the Spanish government with the project MoHuCo PID2020-120049RB-I00. Adria Ruiz acknowledges financial support from MICINN (Spain) through the program Juan de la Cierva.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Adv. Neural Inform. Process. Syst.*, 2018. 6, 8

[2] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *Int. Conf. Learn. Represent.*, 2018. 2

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS One*, 2015. 1, 3

[4] Marianne Bakken, Johannes Kvam, Alexey A Stepanov, and Asbjørn Berge. Principal Feature Visualisation in Convolutional Neural Networks. *ECCV*, 2020. 2

[5] Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[6] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 2018. 1

[7] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.*, 2015. 5

[8] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding Deep Networks via Extremal Perturbations and Smooth Masks. *Int. Conf. Comput. Vis.*, 2019. 2

[9] Ruth C Fong and Andrea Vedaldi. Interpretable Explanations of Black Boxes by Meaningful Perturbation. *Int. Conf. Comput. Vis.*, 2017. 2

[10] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017. 1

[11] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *Int. Conf. Learn. Represent.*, 2019. 2

[12] Shawn Xu Google, A I Healthcare, Subhashini Venugopalan, and Google Research. Attribution in Scale and Space. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2, 5, 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 5

[14] MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys*, 2019. 1

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 5

[16] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. XRAI: Better Attributions Through Regions. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[17] Jinkyu Kim and John Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Int. Conf. Comput. Vis.*, 2017. 1

[18] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *Int. Conf. Learn. Represent.*, 2018. 3

[19] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Adv. Neural Inform. Process. Syst.*, 2017. 2, 3, 6

[20] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 2017. 3

[21] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized Input Sampling for Explanation of Black-box Models. *Brit. Mach. Vis. Conf.*, 2018. 1, 2, 5, 6, 7

[22] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. There and Back Again: Revisiting Backpropagation Saliency Methods. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *International Conference on Knowledge Discovery and Data Mining*, 2016. 2

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 2015. 5

[25] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 2020. 1, 2, 3, 5, 6, 7, 8

[26] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *Int. Conf. Machine Learning*, 2017. 3

[27] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. *34th International Conference on Machine Learning, ICML 2017*, 2017. 5, 7

[28] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Int. Conf. Learn. Represent.*, 2014. 1, 2, 4, 5, 7

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Int. Conf. Learn. Represent.*, 2015. 5

[30] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *Int. Conf. Machine Learning*, 2017. 2, 5, 7

[31] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving For Simplicity: The

All Convolutional Net. *ICLR Workshops*, 2015. 1, 2, 3, 5, 7, 8

[32] Suraj Srinivas and François Fleuret. Full-Gradient Representation for Neural Network Visualization. *Adv. Neural Inform. Process. Syst.*, 2019. 2, 5, 7

[33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. *Int. Conf. Machine Learning*, 2017. 1, 2, 5, 7

[34] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks. *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2020. 2

[35] Huan Xiong, Lei Huang, Mengyang Yu, Li Liu, Fan Zhu, and Ling Shao. On the Number of Linear Regions of Convolutional Neural Networks. *Int. Conf. Machine Learning*, 2020. 2

[36] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. Learning Propagation Rules for Attribution Map Generation. *Eur. Conf. Comput. Vis.*, 2020. 3, 5, 7

[37] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *Eur. Conf. Comput. Vis.*, 2014. 1, 2, 3

[38] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down Neural Attention by Excitation Backprop. *Eur. Conf. Comput. Vis.*, 2016. 3, 5, 6

[39] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 2018. 1

[40] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. 1

[41] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 2

[42] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. *Int. Conf. Learn. Represent.*, 2017. 2