REVIEW

Review of white box methods for explanations of convolutional neural networks in image classification tasks

Meghna P. Ayyar,* Jenny Benois-Pineau, and Akka Zemmari
University of Bordeaux, LaBRI, UMR 5800, Bordeaux, France

Abstract. In recent years, deep learning has become prevalent to solve applications from multiple domains. Convolutional neural networks (CNNs) particularly have demonstrated state-of-the-art performance for the task of image classification. However, the decisions made by these networks are not transparent and cannot be directly interpreted by a human. Several approaches have been proposed to explain the reasoning behind a prediction made by a network. We propose a topology of grouping these methods based on their assumptions and implementations. We focus primarily on white box methods that leverage the information of the internal architecture of a network to explain its decision. Given the task of image classification and a trained CNN, our work aims to provide a comprehensive and detailed overview of a set of methods that can be used to create explanation maps for a particular image, which assign an importance score to each pixel of the image based on its contribution to the decision of the network. We also propose a further classification of the white box methods based on their implementations to enable better comparisons and help researchers find methods best suited for different scenarios. © *2021 SPIE and IS&T* [DOI: 10.1117/1.JEI.30.5.050901]

Keywords: explainable artificial intelligence; deep learning; convolutional neural networks; object classification; interpretability.

Paper 210169V received Apr. 2, 2021; accepted for publication Sep. 3, 2021; published online Sep. 20, 2021.

1 Introduction

Deep learning approaches, which are a part of methods we call today artificial intelligence (AI), have become indispensable for a wide range of applications requiring analysis and understanding of a large amount of data. They can produce promising results that outperform human capabilities in various decision tasks relating to visual content classification and understanding such as face detection,[1] object detection and segmentation,[2] image denoising,[3] video-based tasks such as sports action recognition[4] and saliency detection,[5] and among others. The success of deep learning-based systems in these tasks have also paved the way for their applications to be developed for a variety of medical diagnosis tasks such as cancer detection[6] and Alzhiemer's disease detection[7] on different imaging modalities just to name a few. Along with the usefulness of these tools, the trustfulness and reliability of such systems are also being questioned.

Though the results of deep learning models have been exemplary, they are not perfect, can produce errors, are sensitive to noise in data, and often lack the transparency to have verifiability of the decisions that they make. A specific example is related to the visual task of object classification from an image. The study by Ribeiro et al.[8] showed that the trained network that performed supervised image classification used the presence of snow as the distinguishing feature between the "wolf" and "husky" named classes present in the dataset. Such limitations raise ethical and reliability concerns that need to be addressed before such systems can be deployed and adopted on a wider scale. The objective of explainable AI/deep learning is to design and develop methods that can be used to understand how these systems produce their decisions.

The behavior described in the case of the wolf/husky classification has been termed as the problem of a trained classifier behaving like a "Clever Hans" predictor.[9] Explanation methods aid in the unmasking of such spurious correlations and biases in the model or data and also in

*Address all correspondence to Meghna P. Ayyar, meghna-parameswaran.ayyar@etu.u-bordeaux.fr

understanding of the failure cases of the system. If we can comprehend the reasoning behind the decision of a model, it could also help in uncovering associations that had been previously unobserved, which could aid in furthering the future research trends. It is important to mention that explainability focuses on the attribution of the output based on the input. It does not deal with answering the causality of the features or factors that have led to a decision that has been taken. That is, the explainers are only correlation-based (input–output) and do not make causal inferences.

This study focuses on the task of supervised image classification using specific deep neural network (DNN) architectures, i.e., convolutional neural networks (CNNs). CNNs have become one of the most successful architectures concerning AI tasks relating to images. Hence, the methods presented in the subsequent sections are focused on finding the relation between the predicted output classes and the input features that are the pixels of the image. We specifically focus on white-box methods, which we found quite promising.[10] We also give our viewpoint on the topology of the different explanation methods that have been developed up to now. We present the detailed problem definition in Sec. 3. The following sections present the different explanation methods in detail. In Sec. 7, we discuss different ways of evaluation of explainers and present the comparison of several of them used for the explanation of a well-known VGG-16 CNN[11] for an image classification task. To facilitate reading of this survey, we present this paper organization in Fig. 1.

2 Topology of Explanation Methods for Image Classification Tasks

Samek et al.[12] presented recent trends in the research in explainable AI and some of the directions for future explorations. They have presented a topology for the various explanation methods such as meta-explainers, surrogate/sampling-based, occlusion–based, and propagation-based to name a few. However, with the addition of newer methods and their adaptations to different types of neural networks and datasets, we propose to update the topology based on the domain to which the methods are applied and their inherent design. Comparing recent studies, two major types of explanation methods exist, (i) black-box methods and (ii) white box methods. In this review, for both cases, we mainly focus on the explanations of decisions of trained DNN classifiers. This means that for each sample of the data the methods, we explain the decision of the network. This is why they are called "sample-based" methods.[13] In the following, we will briefly explain the "black box" methods and focus on "white box" methods in image classification tasks.

2.1 *Black Box Methods*

Black box refers to an opaque system. The internal functioning of the model is kept hidden or is not accessible to the user. Only the input and the output of the system are accessible and such methods are termed as black box methods as they are model agnostic.

There are multiple ways to examine what a black-box model has learned.[14] A prominent group of methods is focused on explaining the model as a whole by approximating the black-box model like the neural networks with an inherently interpretable model. One such example is the use of decision trees.[15] Decision trees are human-interpretable as the output is based on a sequence of decisions starting from the input data. To approximate a black-box trained network, Frosst and Hinton[16] have used multiple input–output pairs generated by the network to train a soft decision tree that could mimic the network's behavior. Each node makes a binary decision and learns a filter w and a bias b term, and the probability of the right branch of the tree being selected is given by Eq. (1), where the function $\sigma(x) = 1/1 + e^{-x}$ is the sigmoid logistic function, x is the input, and i is the current node:

$$p_i(x) = \sigma(xw_i + b_i). \tag{1}$$

The leaf nodes learn a simple distribution Q for the different k classes present in the dataset. This method can be qualified as a dataset-based explanation, as the decision tree is built for the whole dataset of pairs input–output.

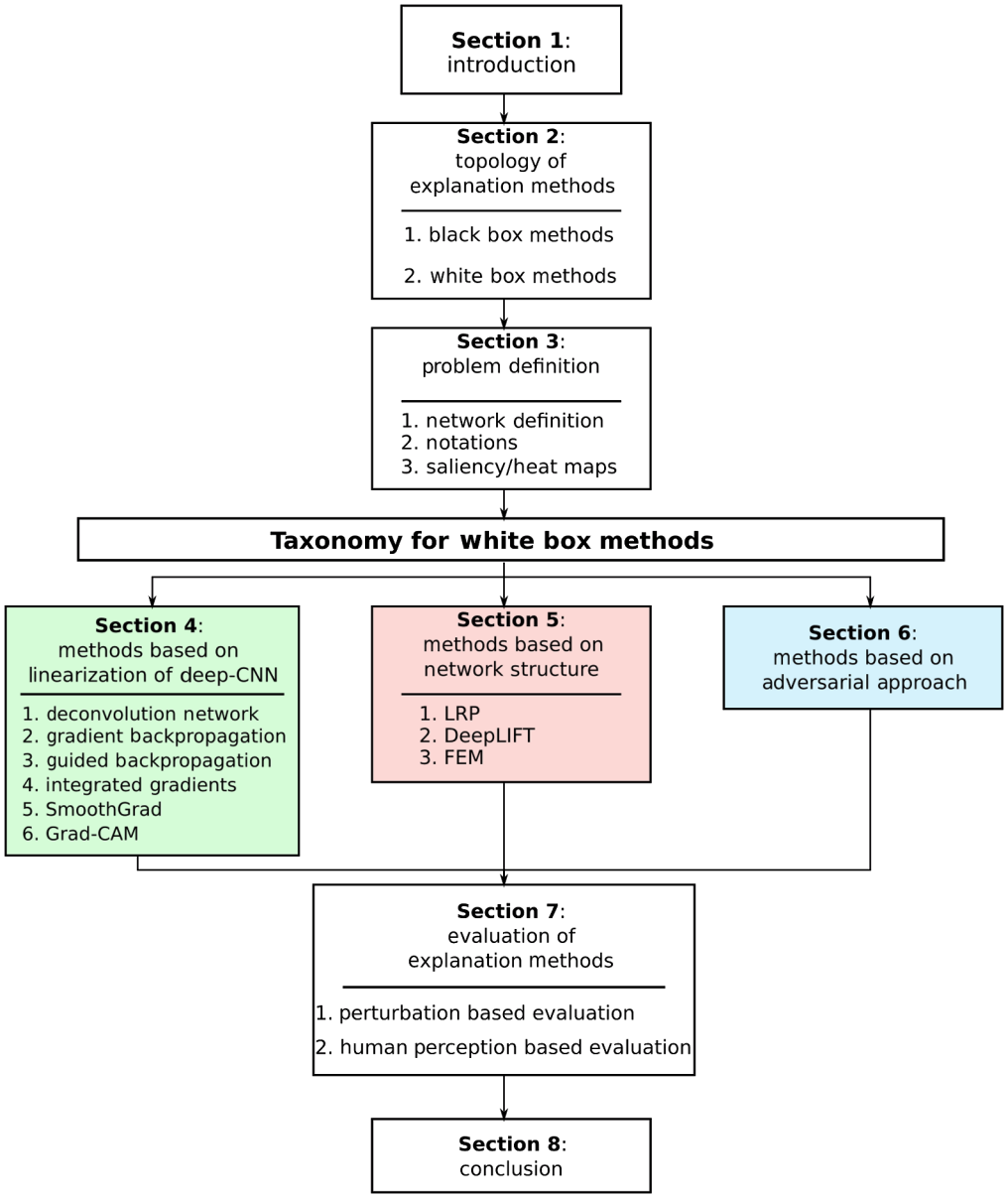

Fig. 1 Organization of this paper.

Sample-based black-box methods deal with explaining a particular output of the model. These methods are not focused on understanding the internal logic of the model for all the classes on a whole but are restricted to explaining the prediction for a single input. The local interpretable model-agnostic explanations[8] method is one such approach that derives explanations for individual predictions. It generates multiple perturbed samples of the same input data and the corresponding outputs from the black box and trains an inherently interpretable model like a decision tree or a linear regressor on this combination to provide explanations.

Taking into account human understanding of visual scenes, such as attraction by meaningful objects in visual understanding tasks, for image classification tasks the regions of the image where the objects are present should have a higher contribution to the prediction. Based on this logic, some methods attempt to occlude different parts of the image iteratively using a sliding window mask.[17] Figure 2 illustrates how the gray-valued window is used for occluding different parts of the images by sliding it across the image. By observing the change in the prediction of the classifier when different regions are hidden, the importance of regions for the final decision is calculated.

Fig. 2 Gray-value sliding window used to occlude to different parts of the image. Image taken from ImageNet.[18]

Fong and Vedaldi[19] also built the explanations on which region contributes to the DNN decision the most by masking. Instead of using a constant gray-value mask, they formulated the explanation as a search for the minimal mask, which changes the classification score for the given image the most. The mask applies a meaningful transformation, which models image acquisition process like blurring. They found the mask by minimizing the expectation of output classification score of the network on the image perturbed using the blurred mask. Instead of using a single mask to perform the search, they apply the perturbation mask stochastically to the image. Also L1 and total variation regularization are used to ensure that the final mask deletes the smallest subset of the image and has a regular structure.

Nevertheless, these methods only aid in identifying if the network is predicting based on a non-intuitive region in the image. The explanations are not useful to identify which layers or filters in a DNN classifier cause these wrong correlations between the input image regions and the prediction. Thus they cannot be used to improve the network performance. Hence, the white box methods, which allow for analyzing internal layers of the network, are more interesting.

2.2 *White Box Methods*

The term "white box" implies a clear box that symbolizes the ability to see into the inner workings of the model, i.e., its architecture and the parameters. Due to the extensive research in DNNs, they are not unknown architectures anymore and studies like Yosinski et al.[20] showed the types of features that are learnt at the different layers in a DNN. Therefore, multiple methods aim to exploit the available knowledge of the network itself to create a better understanding of the prediction and the internal logic of the network thus allowing for further optimization of the architecture and hyperparameters of the model.

In this overview, we propose to deal with the specific case of DNN classifiers such as CNNs. There has been an abundance in the methods proposed to explain the decisions of the CNNs and we need to have a systematic way to compare and understand how they provide the explanations. To do that we propose the following taxonomy for existing "white-box" methods based on their approach used for generating explanations: (i) methods based on linearization of the deep-CNN, (ii) methods based on network structure, and (iii) methods based on adversarial approach. Due to the exploding research in the field, we do not claim to be complete in our taxonomy but believe to have addressed the main trends. The main idea is to have a compact grouping of the different methods so that they can be studied based on the similarity of their approach while also presenting a new group for a method that has a very different approach when compared to others, e.g., the adversarial methods. We present in detail the common characteristics of the methods grouped into a category in further sections.

3 Problem Definition

This section provides the basic terminology and the definitions required to understand the type of network that we will be focusing on, the notations used and how the results are to be visualized.

3.1 *Network Definition*

The problem under consideration is the image classification task. To define the task, first consider a CNN. A simple AlexNet-like[21] CNN is illustrated in Fig. 3. The network consists of a

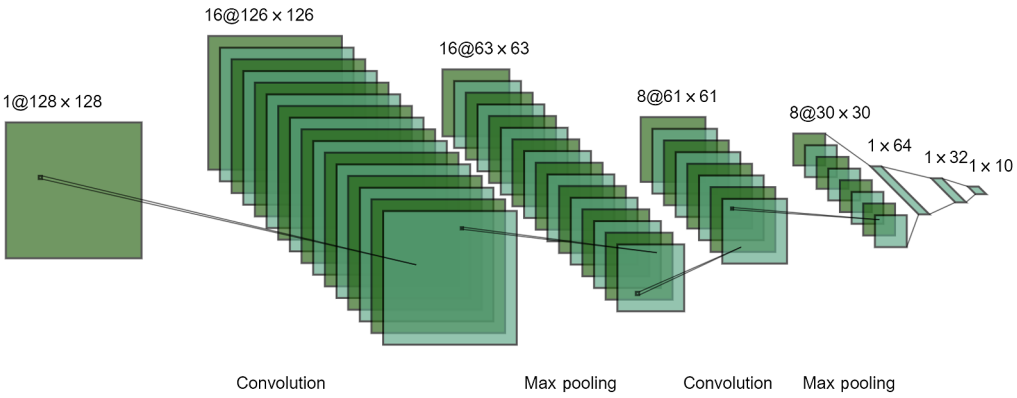

Fig. 3 Architecture of a standard CNN: convolutional layers with non-linear activations, pooling layers, and a perceptron at the end for classification.

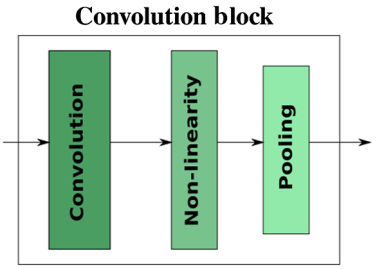

Fig. 4 Structure of a convolution block as proposed by Goodfellow et al.[22]

series of convolutional layers, a non-linear activation layer, and a pooling layer that form the convolution (conv) block as illustrated by Fig. 4. The conv blocks are followed by fully connected (FC) layers that are simple feed-forward neural networks.[22] The rectified linear unit (ReLU) activation shown in Eq. (2) and max pooling are the most commonly used while building CNN classifiers.[23] The last layer of the network has the same dimension as the number of classes in the problem, in the example in Fig. 3, it is 10 implying there are 10 categories of objects to recognize:

$$\mathrm{ReLU}(t) = \max(0, t). \tag{2}$$

3.2 *Notations*

Consider a CNN that takes as input an image x of size $N \times N$ expressed as $x = [x_{11}, \ldots, x_{NN}] \in \mathbb{R}^{N \times N}$ and the output of the classification is a C-dimensional vector $S(x) = [S_1(x), \ldots, S_C(x)] \in \mathbb{R}^C$. Here C represents the number of classes and the image x represents the input features for the network. The scores $S_c(x)$ would be the output classification score for the image x for the class c. The network thus models a mapping $f : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^C$. The output score vector is usually normalized to approximate the probability, thus $\mathbb{R}^C$ is restricted to the [0, 1] interval and the score's vector $S(x)$ sums to 1. The problem of explanations consists in assigning, to each pixel $x_{ij}, i = 1, \ldots, N, j = 1, \ldots, N$, a relevance score $R^c_{ij} \in [0,1]$ with respect to its contribution to the output $S_c(x)$. Otherwise to produce a relevance score map $R^c = [R^c_{11}, \ldots, R^c_{NN}] \in \mathbb{R}^N$ for each of the pixels and/or features of internal convolutions layers of the network to the output $S_c(x)$ where the class c can either be the correct label class or a different class where it can be used to analyze the cause for that classification.

To "explain" pixel importance to the user, a visualization of the scores in R^c is usually performed by computing "saliency/heat maps" and superimposing them on the original image.

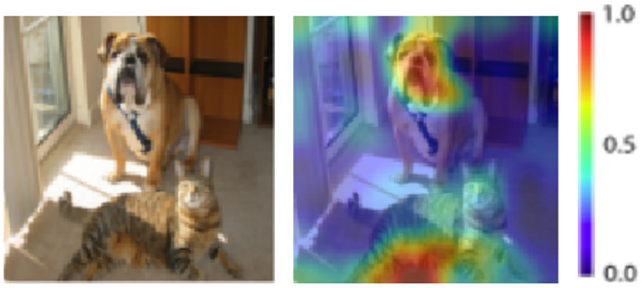

Fig. 5 A saliency map visualization for the sample image. Sample image taken from ImageNet dataset.[18]

3.3 *Saliency/Heat Maps*

A saliency/heat map is the visualization of the relevance score map R^c with color look-up-tables (LUTs) mapping [0, 1] onto a color scale from blue to red as illustrated by Fig. 5. This form of visualizations is necessary for the user to understand and glean insights from the results of the explanation methods. In the current illustration, we have used the "jet" color map that has a linear transition from the maximum value mapping to red, the middle to yellow-green color, and the lowest to blue. Other LUTs can also be used for the visualization of the heat maps, but we have chosen "jet" as it is one of the more popular color maps and is intuitively understandable for a human observer.

Given this kind of network classifier and problem formulation, several methods have been proposed that can be employed for the visualization of relevance score maps given a particular image.

4 Methods Based on Linearization of the Deep-CNN

A (convolutional) neural network is a non-linear classifier. It can be defined as a mapping $f(x)$ from the input (feature space) X to the output score space S. The methods based on the linearization of a CNN produce explanations approximating the non-linear mapping $f(x)$. One of the commonly used approximations is the linear approximation:

$$S_c(x) \approx w^{\mathrm{T}} x + b, \tag{3}$$

where w are the weights, x is the input, and b is the bias related to the network. Different methods employ different ways to calculate the weight and bias parameters of the network approximation and thus produce different explanations.

Methods grouped under this category primarily deal with the calculation of gradients of the network or are equivalent in order to perform the linear approximation of the network. They differ in the layers at which the gradients are computed for example gradient backpropagation does it at the input layer, whereas Grad-CAM does it at the last convolution layer. The following section presents some of these methods with their intuitions and implementation details.

4.1 *Deconvolution Network-Based Method*

The deconvolution network (DeconvNet) proposed by Zeiler and Fergus[17] was a network that reversed the mapping of a CNN. It builds a mapping of the output score S_c to the space of the input pixels x_{ij}. It does not require retraining and directly uses the learned filters of the CNN. Starting with the input image x, a full forward pass through the CNN is done to compute the feature activation throughout the layers. To visualize the features of a particular layer, the corresponding feature maps from the CNN layer are passed onto the DeconvNet. In the DeconvNet, the three steps: (i) unpooling (ii) rectification, and (iii) filtering are done at each layer iteratively till we reach the input features layer.

- *Unpooling*. Max pooling operation in a CNN is non-invertible. Hence to reverse this, during the forward pass in the network, at each layer, the maxima locations are saved to a matrix called the "max location switches." During the unpooling, the values from the previous layers are mapped to only the locations of maxima and the rest of the positions have a 0 assigned to them.
- *Rectification*. After applying the unpooling, an ReLU function [Eq. (2)] is applied onto the matrix to ensure that only positive influences on the output are backpropagated.
- *Filtering*. This operation is the inverse operation to the convolution in the forward pass. To achieve this, the DeconvNet convolves the rectified maps with the vertically and horizontally flipped version of the filter that has been learned by that layer in the CNN. The authors show that filters thus defined from learnt CNN filters are the deconvolution filters. We show the mathematical derivation of this in Appendix A.

Performing these operations iteratively from the layer of our choice to the input pixel layer helps to reconstruct the features from the layer that correspond to different regions in the input image x. The importance of pixels is then expressed with a heat map.

4.2 *Gradient Backpropagation*

The gradient backpropagation method[24] was proposed to explain prediction of a model based on its locally evaluated gradient. The local gradient of the output classification score S_c with respect to the input x at a particular image x_0 is used to calculate the weights parameter w from Eq. (3). This means that the linear approximation of the non-linear mapping $f(x)$ is formulated as a Taylor first-order expansion of $f(x)$ in the vicinity of a particular image x_0 and $b = f(x_0)$. The weight parameters are thus calculated as in the following equation:

$$w = \frac{\partial S_c}{\partial x}|_{x_0}. \tag{4}$$

The partial derivative of the output classification score with respect to the input corresponds to the gradient calculation for a single-backpropagation pass for a particular input image x. It is equivalent to the backpropagation step that is performed during training, which usually corresponds to a batch of images. For this method, the notation of gradient at x_0 is to show that the backpropagation is for just one image. Also during the training of a CNN, the backpropagation step stops at the second layer of the network for efficiency as the aim is not to change the input values. But with this method, the backpropagation is performed till the input layer to inspect which pixels affect the output the most.

The final heat map relevance scores R_{ij} for a particular pixel i, j in the input 2D image are calculated as shown in Eq. (5) in the case of a gray-scale image:

$$R_{ij} = \left\| \frac{\partial}{\partial x_{ij}} S_c \right\|. \tag{5}$$

For an RGB image, the final map is calculated as the maximum weight of that pixel from the weights matrices from each of the three channels as shown in Eq. (6), where k corresponds to the different channels in the image:

$$R_{ij} = \max_k |w(i, j, k)|. \tag{6}$$

Also these gradients can be used for performing a type of sensitivity analysis. The magnitude of the derivatives that have been calculated could be interpreted to indicate the input pixels to which the output classification is the most sensitive. A strong gradient magnitude value would correspond to the pixels that need to be changed the least to affect the final class score the most.

Simonyan et al.[24] also showed that the gradient backpropagation is a generalization of DeconvNet (Sec. 4.1). Indeed, this can be shown by comparing the three operations that DeconvNet performs with the gradient calculation.

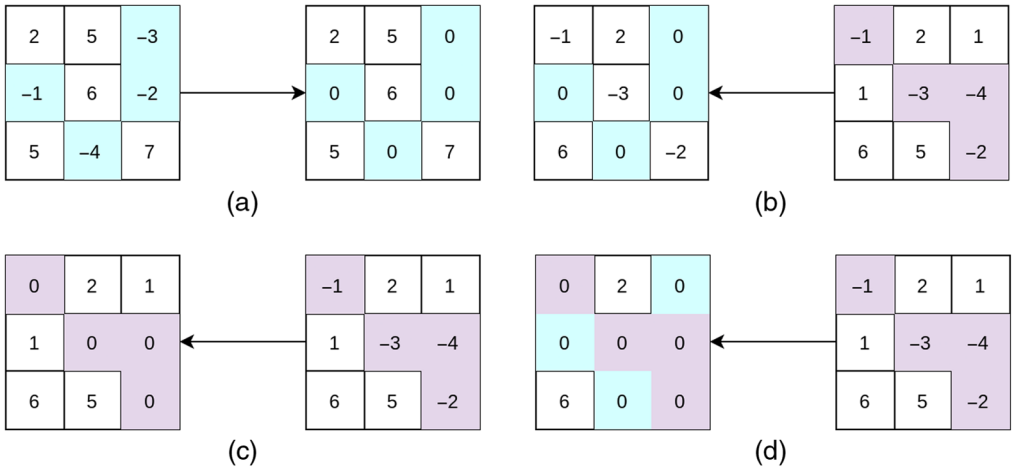

Fig. 6 The ReLU operation during (a) forward pass, (b) backpropagation, (c) backpropagation with DeconvNet, and (d) guided backpropagation.

- *Unpooling*. During basic backpropagation at a max-pooling layer, the gradients are back-propagated to only those positions that had the max values during the forward pass. This is exactly the same operation that is achieved by the use of the max location switches matrix in the DeconvNet (see Sec. 4.1).
- *Rectification*. For a CNN with the output of a convolution layer n as Y, the application of the ReLU activation is performed as $Y_{n+1} = \max(Y_n, 0)$, where Y_{n+1} then is the input for the next layer in the network. During the gradient backpropagation, the rectification applied on the gradient map is based on the input, i.e., on those position where $Y_n > 0$. Whereas, in the DeconvNet, the rectification is applied on the unpooled maps and hence corresponds to the condition of $Y_{n+1} > 0$. Figures 6(b) and 6(c) show the changes in the calculation of the two matrices based on this difference in the operations.
- *Filtering*. As shown in Appendix A, the vertical and horizontally flipped filter that is used during the filtering step in the DeconvNet corresponds to the gradient calculation of the convolution with respect to the input x. This is the same step as the gradient backpropagation method performs and hence this step is equivalent for the two methods.

Except for the rectification step, the two methods are equivalent in their calculations and therefore, the gradient backpropagation method can be seen as a generalization of the DeconvNet.

4.3 *Guided Backpropagation*

Computing a saliency map based on gradients gives an idea of the various input features (pixels) that have contributed to the neuron responses in the output layer of the network. The primary idea proposed by Springenberg et al.[25] is to prevent the backpropagation of negative gradients found in the deconvolution approach as they decrease the activation of the higher layer unit we aim to visualize. This is achieved by combining the rectification operation performed by the DeconvNet and the gradient backpropagation. As shown in Fig. 6(d), guided backpropagation proposes to restrict the flow of the gradients that have a negative value during backpropagation and also those values that had a negative value during the forward pass. This nullification of negative gradient values is called the "guidance." Using the guidance step results in sharper visualizations for the descriptive regions in the input image.[25] Figure 7 shows the heat maps generated by the gradient backpropagation and guided backpropagation methods for the network trained on ResNet34 architecture.[27] It can be seen that the guidance steps result in reducing the number of pixels that have a higher importance score and hence produce slightly sharper visualization.

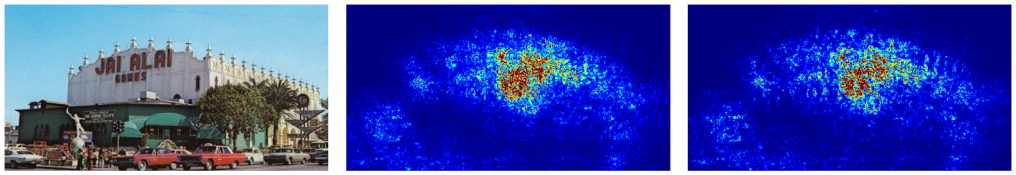

Fig. 7 Samples showing the saliency maps for the (a) sample image, (b) gradient backpropagation, and (c) guided backpropagation methods. Sample taken from MexCulture Architectural Styles dataset.[26]

4.4 *Integrated Gradients*

Sundararajan et al.[28] proposed to calculate the saliency map as an integration of the gradients for a set of images that are created from the transformation of a baseline image x' to the input image x. They propose the baseline as a black image, then the series of images is produced by a linear transformation $x(\alpha) = x' + \alpha \times (x - x')$. Thus if we denote by x_i the value of i-feature in our x, then Eq. (7) shows the calculation of the integrated gradients for the network with output classification for a class c as $S_c(x)$. This forms the map of relevance scores R^c with a corresponding score for each input pixel x_i:

$$R_i^c = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial S_c(x' + \alpha \times (x - x')}{\partial x_i} \mathrm{d}\alpha. \tag{7}$$

The parameter α varies in [0, 1] and the term within the partial derivative would go from the baseline image to the final input as we integrate over α as shown in Fig. 8. In practice, the integral is approximated by a summation over a fixed number of samples, i.e., Riemann approximation.

The authors observed that if there were slight changes in the pixel value of the image x such that visually the image did not appear to have changed, the gradients calculated by the gradient backpropagation methods showed large fluctuations in their values. For a small amount of noise present in the image, the visualization with gradient backpropagation was different to that of the original image. In the integrated gradients method, an averaging is performed over a sample of images and so the final relevance maps would be less sensitive to these fluctuating gradient values when compared with the other gradient-based methods.

4.5 *SmoothGrad*

An alternative method to circumvent the issue of noisy saliency maps called the SmoothGrad was proposed by Smilkov et al.[29] The idea of this method is to have a smoother map with sharper visualizations by averaging over multiple noisy maps. To achieve this, the authors propose to add a small noise sampled from Gaussian distribution $\mathcal{N}(0, \sigma^2)$, where σ is the standard deviation, to the input image x for each color channel. Thus they create n samples of the input image with a small amount of noise added to its pixels. The relevance score maps are calculated for each of these images and the average of these n generated maps gives the final relevance score map for the image x as shown in Eq. (8). SmoothGrad is not a standalone method rather it can be used as

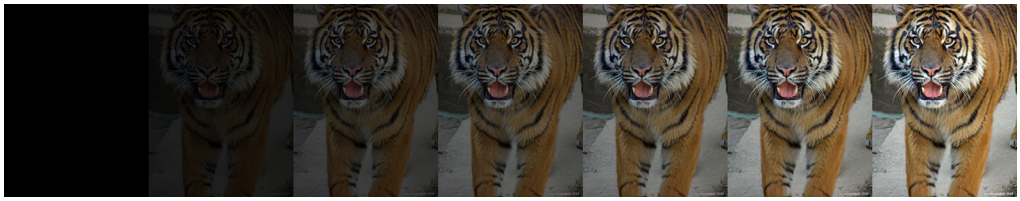

Fig. 8 Transformation of the baseline image for integrated gradients for 7 values of $\alpha = [0,1]$. Image has been taken from ImageNet database.[18]

an extension to other gradient-based methods to reduce the visual noise of the saliency maps. The authors observe that adding about 10% to 20% noise to sampled images produced sharper maps. The parameter σ was chosen such that $\sigma/x_{\max} - x_{\min}$ was in the range of $[0.1, 0.2]$. Here $x_{\max}$ and $x_{\min}$ refer to the maximum and minimum values of the pixels of the image:

$$x_i = x + \mathcal{N}(0, \sigma^2), \quad \hat{R}^c(x) = \frac{1}{n}\sum_{i=1}^{n} R^c(x_i). \tag{8}$$

4.6 *Gradient-Class Activation Mapping*

Gradient class activation mapping (Grad-CAM)[30] is a *post hoc* explanation via visualization of class discriminative activations for a network. Similar to gradient-based methods, Grad-CAM leverages the structure of the CNN to produce a heat map of the pixels from the input image that contribute to the prediction of a particular class.

A key observation that Grad-CAM relies on is that the deeper convolutional layers of a CNN act as high-level feature extractors.[31] So the feature maps of the last convolution layer of the network would contain the structural spatial information of objects in the image. Therefore, instead of propagating the gradient till the input layer like other gradient-based methods, Grad-CAM propagates the value from the output till the last convolutional layer of the network.

The features maps from the last convolution layer cannot be used directly as they would contain information regarding all the classes present in the dataset. Assuming that the last convolution layer of the network has k feature maps named $A^1, A^2, \ldots, A^k$, the Grad-CAM method proposed to determine an importance value for each of the k maps to the class c predicted by the network. This value is calculated as the global average pooling of the gradient of the classification score $S_c(x)$ with respect to the activation values A^k for that feature map. As shown in Eq. (9), α_k^c is the importance value for the feature map k and there are k such weights that are computed:

$$\alpha_k^c = \frac{1}{Z}\sum_i \sum_j \frac{\partial S_c(x)}{\partial A_{ij}^k}, \tag{9}$$

Here $Z = h \times w$, where h and w correspond to the height and width dimension of each feature map.

The α_k^c weights are then used to weight each feature in the k feature maps, the latter are then averaged. This gives us the relevance score map R^c, and an ReLU (rectification) function is applied over this map [see Eq. (10)] to nullify the features that are negative and retain only those values that have a positive influence. At this stage, the relevance map R^c is a 2D map with the same spatial dimension as the feature maps of the last convolution layer. To have a correspondence to the input image x, R^c is upsampled to the spatial dimension of x using interpolation methods and scaled to the interval of $[0,1]$ to visualize the final heat map:

$$R^c = \mathrm{ReLU}\left(\sum_k \alpha_k^c A^k\right). \tag{10}$$

Grad-CAM is the generalization of previously proposed by Zhou et al.[32] CAM method, which requires squeezing of the feature maps of the last conv layer by average pooling to form the input of the FC layer in the network. Grad-CAM on the contrary can be applied to all the architectures of deep CNNs.

4.6.1 *Guided Grad-CAM*

The heat maps produced by Grad-CAM are coarse, unlike the other gradient-based methods.[30] As the feature map of the last convolutional layer has a smaller resolution compared to the input image x, Grad-CAM maps do not have fine-grained details that are generally seen in other gradient-based methods. To refine the maps, a variant of the method called the guided Grad-CAM has been proposed, which is a combination of the Grad-CAM and the guided backpropagation

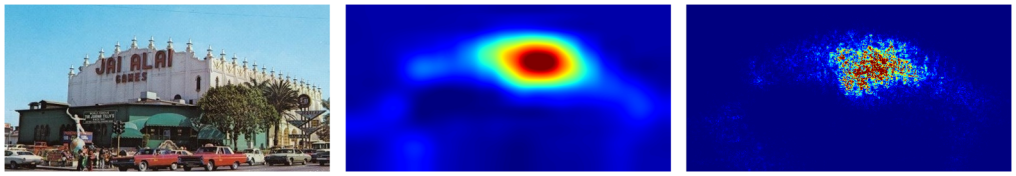

Fig. 9 Samples showing the saliency maps for the (a) sample image, (b) Grad-CAM, and (c) guided Grad-CAM methods. Image taken from MexCulture Architectural Styles dataset.[26]

method by doing an element-wise multiplication of the two maps. The heat map that is obtained by this operation has been observed to have a higher resolution.[30] We illustrate maps obtained by Grad-CAM and guided Grad-CAM in Fig. 9.

5 Methods Based on Network Structure

This category of methods integrates the architecture of the network while explaining the output. Starting from an output neuron, they employ different local redistribution rules to propagate the prediction to the input layer to obtain the relevance score maps. Unlike the previous category, these methods do not compute gradients in the network. In this section, we present the details of them.

5.1 *Layer-Wise Relevance Propagation*

Layer-wise relevance propagation (LRP) is an explanation method proposed by Bach et al.[33] that explains the decision of a network for a particular image by redistributing the classification score S_c for a class c backward through the network. The method does not use gradient calculations but defines the activation of the output neuron (either the predicted class or another class that is being considered) as the relevance value and a set of local rules for the redistribution of this relevance score backward till the input, layer by layer. The first rule that they propose is that of "relevance conservation." Let the neurons in the different layers of the network be denoted by ν, ξ, o etc. and $S_c(x)$ the classification score for the input image x regarding the class c. Then according to the relevance conservation rule, the sum of the relevance scores of all the neurons in each layer is a constant and equals $S_c(x)$ as shown in the following equation:

$$\sum_{\nu} R_{\nu} = \ldots = \sum_{\xi} R_{\xi} = \sum_{o} R_{o} = \ldots = S_c(x). \tag{11}$$

Let l and $l+1$ be two consecutive layers in the network and i, j denote the neurons belonging to these layers, respectively. The relevance of the neuron j based on $S_c(x)$ can be written as R_j^c. If neuron i is connected to neuron j then it is assigned a relevance value of R_j^c weighted by the activation of the neuron a_i and the weight of the connection between the two neurons w_{ij}. Similarly, neuron i receives a relevance value from all the neurons that it is connected to in the next layer $(l+1)$. The sum of all the relevance contributions that the neuron receives from the neurons it was connected to in the next layer is the final relevance value R_i^c that is assigned to the neuron as shown in Eq. (12). The denominator term in Eq. (12) is the normalization value that is used to ensure relevance conservation rule Eq. (11). This rule is termed as the LRP-0 rule:[34]

$$R_i^c = \sum_j \frac{a_i w_{ij}}{\sum_{0,q} a_q w_{qj}} R_j^c. \tag{12}$$

In this equation, the summation $\sum_{0,q} a_i w_{qj}$ is done for all the neurons in the lower layer $q = 0, \ldots, Q_l$ including the bias neuron in the network. The activation of the bias neuron is considered as $a_0 = 1$ and the weight of the connection is denoted as w_{0j}. For the relevance

propagation, the bias neuron is considered only for this term and is not considered elsewhere. Note that the authors propose these rules only for the specific case of rectifier networks, i.e., for networks with ReLU as the non-linearity. The relevance of the output neuron is considered as its activation taken before the Softmax layer.

Similarly, there exist a few other rules that improve on the LRP-0 rule for the propagation of relevance as presented in the following list.

- *Epsilon rule (LRP-ε)*. To improve the stability of the LRP-0 rule, a small positive term ε is added to the denominator as shown in Eq. (13). The ε term also reduces the flow of the relevance if the activation of the neuron is very small or there is a weak connection between the two neurons. If the value of ε is increased, then it aids in ensuring only the stronger connections receive the redistributed relevance:[34]

$$R_i^c = \sum_j \frac{a_i w_{ij}}{\sum_{0,q} a_q w_{qj} + \varepsilon} R_j^c. \tag{13}$$

- *LRP- γ*. The parameter γ was introduced to improve the contributions of the connections that had a positive weight (w_{ij}^+) as shown in Eq. (15). The function $(f)^+ = \max(0, f)$ and so the neurons with a positive weight connection receive a higher relevance score during propagation:

$$R_i^c = \sum_j \frac{a_i \cdot (w_{ij} + \gamma w_{ij}^+)}{\sum_{0,q} a_q \cdot (w_{qj} + \gamma w_{qj}^+)} R_j^c. \tag{14}$$

- *LRP $\alpha\beta$ rule*. Two parameters α and β are used to control separately the positive and negative contributions to the relevance propagation. The function $(f)^+ = \max(0, f)$ and $(f)^- = \min(0, f)$ and the parameters are constrained under the rule of $\alpha = \beta + 1$:

$$R_i^c = \sum_j (\alpha \frac{(a_i w_{ij})^+}{\sum_{0,q} (a_q w_{qj})^+} - \beta \frac{(a_i w_{ij})^-}{\sum_{0,q} (a_q w_{qj})^-}) R_j^c. \tag{15}$$

5.1.1 *LRP as deep Taylor decomposition*

Montavon et al.[35] proposed a framework to connect a rule-based method like LRP and the Taylor decomposition method as a way to theoretically explain the choice of the relevance propagation rules.[33] They proposed a method called the deep Taylor decomposition (DTD), which treats LRP as consecutive Taylor expansions applied locally at each layer and neuron. The main idea that DTD uses is that a deep network can be written as a set of subfunctions that relate the neurons of two consecutive layers. Instead of treating the whole network as a function f, DTD expresses LRP as a series of mapping of neurons i at a layer l to the relevance R_j of the neuron j in layer $l + 1$.

The Taylor expansion of the relevance score R_j can be expressed as function of the activations a_i at some "root" point $\tilde{\mathbf{a}}$ in the space of the activations as shown in the following equation:

$$R_j(\mathbf{a}) = R_j(\tilde{\mathbf{a}}) + \sum_{0,i} (a_i - \tilde{a}_i)[\nabla R_j(\tilde{\mathbf{a}})]_i + \ldots \tag{16}$$

The first-order term in this expansion can be used to determine how much of the relevance R_j is redistributed to the neurons in the lower layer. The main challenge in the computation of the Taylor expansion, in this case, is that of finding the appropriate root point $\tilde{\mathbf{a}}$ and computes the local gradients.

To compute the function $R_j(\mathbf{a})$, the authors propose to substitute it with a relevance model that is simpler to analyze. From the relevance propagation rules of LRP (Sec. 5.1), it can be seen that the relevance score of a neuron can be written as function of its activations as $R_i = a_i \cdot r_i$,

where r_i in the case of the LRP-0 rule [Eq. (12)] can be written as $r_i = \sum_i \frac{w_{ij}}{\sum_{0,i} a_i w_{ij}} R_j$. As the LRP rules are described for deep rectifier networks, the relevance function $R_j(\mathbf{a})$ is expressed based on the ReLU activation as

$$R_j(\mathbf{a}) = \max(0, \sum_{0,i} a_i w_{ij}) \cdot r_j. \tag{17}$$

A Taylor expansion of this function gives

$$R_j(\mathbf{a}) = R_j(\tilde{\mathbf{a}}) + \sum_{0,i} (a_i - \tilde{a}_i) \cdot w_{ij} r_j. \tag{18}$$

Due to the linearity of the ReLU function on the domain of positive activations, the higher order terms in the expansion are zero. The choice of the root point would ensure that the zero-order terms can be made small. The first-order term computation is fairly straightforward and would identify how much of the relevance value R_j should be redistributed to the neurons of the lower layer. Different LRP rules that have been presented previously can be derived from Eq. (18) based on the choice of the reference point $\tilde{\mathbf{a}}$. For instance, LRP-0 rule shown in Eq. (12) can be derived by choosing $\tilde{\mathbf{a}} = 0$ and LRP-ε as shown in Eq. (13) by choosing $\tilde{\mathbf{a}} = \varepsilon \cdot (a_j + \varepsilon)^{-1}$.

5.2 *Deep Learning Important Features*

The primary idea of DeepLIFT, a method proposed by Shrikumar et al.,[36] is similar to the LRP method explained in Sec. 5.1. The major difference between the two methods is that DeepLIFT establishes the importance of neurons in each layer in terms of the difference of their response to that of a "reference state." The reference state is either a default image or is an image chosen based on domain-specific knowledge. This reference could be an image that has the specific property against whose differences in the explanations are meant to be calculated. For example, it could be a black image in the case of the MNIST dataset as the backgrounds of the images in that dataset are all black. DeepLIFT aims to explain the difference in the output produced by the input image and the output of the reference state based on the difference between the input image and the chosen reference image.

For the output classification score S_c of the input image x and the output score of reference state as S_c^0, the difference term ΔS_c is defined as $S_c - S_c^0$. For the neurons $y_1, \ldots, y_i, \ldots, y_n$ belonging to a layer, the relevance is denoted by $R_{\Delta y_i, \Delta S_c}$. Δy_i denotes the difference between the activations of the neuron y_i for the input image and the reference state. Similar to the LRP method, DeepLIFT has the "summation to delta" rule where the summation of the relevance of neurons at each is constant and equal to ΔS_c as shown in the following equation:

$$\sum_{i=1}^{n} R_{\Delta y_i, \Delta S_c} = \Delta S_c. \tag{19}$$

In order to explain the propagation rules, the authors define the term multiplier: $m_{\Delta y \Delta S_c}$, which is defined as the contribution $R_{\Delta y, \Delta S_c}$ of the difference in the reference and input image activations Δy to the difference in the output prediction ΔS_c divided by Δy as shown in the following equation:

$$m_{\Delta y \Delta S_c} = \frac{R_{\Delta y \Delta S_c}}{\Delta y}. \tag{20}$$

The multiplier is a term that is similar to a partial derivative but defined over finite differences.[36] They also define the "chain rule for multipliers" similar to the chain rule used with derivatives as shown in Eq. (21), where z_j denotes the neurons in intermediate layers between the neurons y and the output S_c:

$$m_{\Delta y_i \Delta S_c} = \sum_j m_{\Delta y_i \Delta z_j} m_{\Delta z_j \Delta S_c}. \tag{21}$$

Similar to LRP, the authors also separate the relevance values into two terms: positive and negative as they can then be treated differently if required. For each neuron y, the two terms Δy^+ and Δy^- are the positive and negative components, respectively. These components can be found by grouping the positive and negative terms that contribute to the calculation of Δy. Based on this idea, the difference in the neuron activations of the input image and the reference state, and the relevance contribution can be written as shown in the following equation:

$$\Delta y = \Delta y^+ + \Delta y^-, \quad R_{\Delta y \Delta S_c} = R_{\Delta y^+ \Delta S_c} + R_{\Delta y^- \Delta S_c}. \tag{22}$$

Using these terms, DeepLIFT proposes three rules that can be applied to a network for different layers to propagate the relevance from the output to the input layer.

- *Linear rule*. The linear rule is applied for the FC and convolution layers (not applicable for the non-linearity layers). Considering the function $z = \sum_i w_i y_i + b$, where z is the activation of the neuron in the next layer, y_i are the activations to the neuron from the previous layer, and w_i is weights of the connections, then we have $\Delta z = \sum_i w_i \Delta y_i$ based on the difference taken with the activations of the reference state neurons. The relevance contribution is then written as $R_{\Delta y_i \Delta z} = w_i \Delta y_i$. The multiplier in this case is given by $m_{\Delta y_i \Delta z} = w_i$.
- *Rescale rule*. The rescale rule is applied to layers with the non-linearities like the ReLU. Consider the neuron z to be the non-linear transformation of y as $z = g(y)$. In the case of the ReLU, function [Eq. (2)] denoted by $g(y)$. Considering the summation to delta property, the relevance contribution is $R_{\Delta y \Delta z} = \Delta z$ as there is only one input y. Hence, the multiplier in this case would be $m_{\Delta y \Delta z} = \Delta z / \Delta y$.
- *Revealcancel rule*. The revealcancel rule treats the positive and negative contributions to relevance values, separately. The impact of the positive and negative components of y given as Δy^+ and Δy^- on the components of z given by Δz^+ and Δz^- are calculated, separately. Instead of the straightforward calculation, the value of Δz^+ is computed as the average of two terms. The first term is the average impact of the addition of only Δy^+ terms on the output of the nonlinearity g. With y^0 as the value of the reference state at that neuron, the impact of Δy^+ is calculated by comparing the difference in the function value when it is included on top of y^0 and is given as $g(y^0 + \Delta y^+) - g(y^0)$. The second term computes the impact of Δy^+ after the negative terms have been included. So the term $g(y^0 + \Delta y^+ + \Delta y^-) - g(y^0 + \Delta y^-)$ measures the impact of Δy^+ when both the reference and negative terms are present. This computation is shown in the following equation:

$$\Delta z^+ = \frac{1}{2}(g(y^0 + \Delta y^+) - g(y^0)) + \frac{1}{2}(g(y^0 + \Delta y^+ + \Delta y^-) - g(y^0 + \Delta y^-)). \tag{23}$$

Similarly for the calculation of Δz^-, the average individual Δy^- term is first computed in the absence of the positive term Δy^+ and then another term with the inclusion of Δy^+ is added to get the total impact as shown in the following equation:

$$\Delta z^- = \frac{1}{2}(g(y^0 + \Delta y^-) - g(y^0)) + \frac{1}{2}(g(y^0 + \Delta y^- + \Delta y^+) - g(y^0 + \Delta y^+)). \tag{24}$$

Thus two multipliers that will be computed using this rule are as shown in Eq. (25), where Δz^+ and Δz^- are calculated using Eqs. (23) and (24), and Δy^+ and Δy^- correspond to the sum of the positive and negative terms of Δy:

$$m_{\Delta y^+\Delta z^+} = \frac{R_{\Delta y^+\Delta z^+}}{\Delta y^+} = \frac{\Delta z^+}{\Delta y^+}, m_{\Delta y^-\Delta z^-} = \frac{R_{\Delta y^-\Delta z^-}}{\Delta y^-} = \frac{\Delta z^-}{\Delta y^-}. \tag{25}$$

The authors propose that the relevance scores that have been assigned to Δy^+ and Δy^- are then distributed to the input features using the linear rule.

5.3 *Feature-Based Explanation Method*

Feature-based explanation method (FEM) proposed by Fuad et al.,[37] similar to Grad-CAM, employs the observation that deeper convolutional layers of the network act as high-level feature extractors.

Let us consider a CNN that comprises a single Gaussian filter at each convolution layer. Then the consecutive convolutions of the input image x with the Gaussian filters followed by the pooling (downsampling) would be the same operation that would be performed to create a multi-resolution Gaussian pyramid. In this pyramid, the image at the last level would have only the spatial information of the main objects, which are present in the image. Considering a standard CNN, the learned filters at the deeper convolution layers behave similar to the high pass, i.e., derivative filters on the top of Gaussian pyramid (some examples are given in Ref. 23). This would imply that the information contained in the feature maps of the last convolution layer correspond to the main object that has been detected by the network from the given input image x.

Hence, FEM proposes that the contribution of the input pixels for the network decision can be directly inferred from the features of x that have been detected at the last conv layer of the network. Also FEM proposes that the final decision of the output would be influenced by the "strong" features from k maps in the last conv layer. FEM supposes that the k feature maps of the last convolutional layer have a Gaussian distribution. In this case, the strong features from these maps would correspond to the "rare" features. The authors propose a K-sigma filtering rule to identify these rare and strong features. Each of the feature maps in layer k is thresholded based on the K-sigma rule to create binary maps $B_k(a_{i,j,k})$, where i, j denote the spatial dimension of the k feature maps as shown in Eq. (26). The mean μ_k and standard deviation σ_k are calculated for each of the k feature maps followed by the thresholding to create k binary maps (B_k). K is the parameter that controls the threshold value and is set to 1 by the authors in their work:

$$B_k(a_{i,j,k}) = \begin{cases} 1, & \text{if } a_{i,j,k} \geq \mu_k + K * \sigma_k \\ 0, & \text{otherwise} \end{cases}. \tag{26}$$

The hyperparameters of a DNN such as the number of filters to train at each layer is often set arbitrarily, as the hyperparameter optimization process is a heavy computational problem. Hence, channel attention mechanisms have been proposed in the DNNs[38] to improve classification accuracy. These mechanisms select important feature channels (maps) in an end-to-end-training process. Inspired by these models, the authors hypothesize that not all feature maps will be important for the classification. The importance is always understood as the magnitude of positive features in the channels. Hence, a weight term equal to the mean of the initial feature maps $w_k = \mu_k$ is assigned to each of the binary maps B_k. The importance map R is computed as the linear combination of all the weighted binary maps B_k and then normalized to the interval [0, 1]. The importance map R is upsampled to the same spatial resolution as the input image x by interpolation.

FEM eliminates the need to calculate the gradients from the output neuron and provides a faster and simpler method to get an importance score of the input pixels based only on the features that have been extracted by the network. It does not examine the classification part of the network but uses only the feature extraction part of the CNN to explain the important input pixels that have been extracted by the network to produce the decision. The method is applicable both for 2D and 3D images or video, considered as a 2D $+t$ volume. We will now illustrate it in the problem of image classification from ImageNet database performed with the VGG16.[11] We propose the reader to visually compare the heat maps presented in Fig. 10 obtained using different LRP rules[33] and FEM.

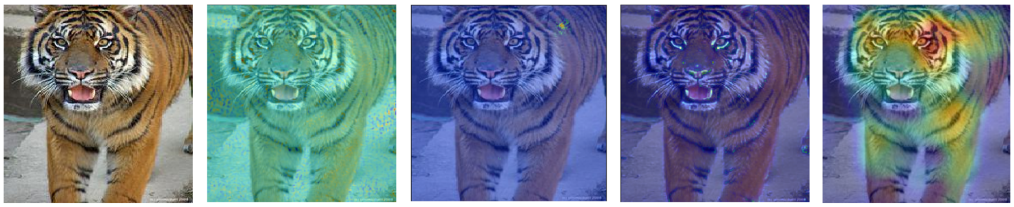

Fig. 10 Explanation heat maps obtained for (a) sample image, (b) LRP-ε, (c) LRP-ε ignore bias, (d) LRP-$\alpha\beta$ with $\alpha = 1$, $\beta = 0$, and (e) FEM.

It can be seen from this figure that LRP heat maps are dependent on the rule that is used. In this case, the LRP-ε assigns equal weight to both positive and negative features, and in Fig. 10(a), most of the input pixels get assigned a higher relevance score. LRP-ε without bias from Fig. 10(c) results in a heat map that has the importance only in a smaller region of the tiger near the top. Without the added bias term, the relevance scores are not distributed properly to the other regions. The $\alpha\beta$ rule with $\alpha = 1$ and $\beta = 0$ only considers the features with a positive influence. As illustrated in this figure, this rule only highlights the contours with higher contrast in the image. Though FEM also considers the positive features (features are taken after the ReLU), the heat map is more holistic and highlights the important regions in the image.

6 Methods Based on Adversarial Approach

The adversarial approach for explanations is more recent and differs in its principles from previously presented categories. They are taken from the principles of adversarial attacks and the use of generative networks like generative adversarial networks (GANs) to explain the DNN classifiers. Thus adversarial methods deserve a separate category.

Many recent works have used adversarial attacks on CNNs to demonstrate the susceptibility of the networks to simple methods that could lead the network to make completely wrong predictions.[39] Different adversarial attacks on the network can be used to interpret its behavior.[40] Sample images that produce adversarial results give hints about the behavior of the network.[41,42] For example, the one-pixel attack proposed by Su et al.[43] showed that the network can have a completely wrong prediction when just one pixel in input image has been changed. Studying these adversarial attacks, we can interpret the regions of the image that the network focuses on to make a decision.

In addition to the adversarial attack-based interpretation of the network, we bring focus on a recent adversarial learning-based explanation method proposed by Charachon et al.[44] that uses a GAN[45]-based model. GAN is a type of network architecture with two components: (i) generator and (ii) discriminator, which simultaneously trains the generator (G) to learn the data distribution and the discriminator (D) to estimate if the sample belongs to the dataset or has been generated by G.

For the case of a binary classification task on medical images, the authors[44] along with their CNN classifier use two generator networks (i) similar image generator $\bar{g}_s$ and (ii) adversarial image generator $\bar{g}_a$ to produce explanations. The network $\bar{g}_s$ is trained to generate an image that has the same output as the input image x by the network. The $\bar{g}_a$ network is trained to produce an image with a prediction that is opposite to that of the x (adversarial). Consequently, the authors propose that the difference in these two generated images forms the explanation of the output by the network. For a given image x, the explanation of the classifier is then given as shown in the following equation:

$$R(x) = |\bar{g}_s(x) - \bar{g}_a(x)|. \tag{27}$$

The approach of just one network to generate an adversarial image (x_a) and use the difference in the input image x and the x_a as the explanation was observed to produce noisy and

non-intuitive features. To improve this, the authors use the two-generator approach and train $\bar{g}_s$ and $\bar{g}_a$ to sample from the same adversarial space, to have minimal differences in their learnt parameters but produce images with opposite classifications by the CNN.

7 Evaluation of Explanation Methods

The evaluation of explanation methods remains an open research question. Today, we can observe two trends in the evaluation of explanation methods. The first one is perturbation-based when the input, dataset, or network is perturbed and the induced variation in the importance maps serves as an explainer. The second one is an attempt to correlate the explainer with the human understanding of visual scenes and assess it with user feedback. For comparison of explanation maps, usual metrics for saliency maps are computed[46] such as Spearman correlation coefficient (SCC), Pearson correlation coefficient (PCC), structural similarity index measure (SSIM), and L1 similarity to name a few.

7.1 *Perturbation-Based Evaluation*

Samek et al.[47] argued that heatmaps are representative of a classifier's view and will not necessarily have to follow human intuition or highlight the primary object in the image. Hence, to assess the relevance of the heat maps, they order the pixels in increasing importance score associated with them and then iteratively perturb the pixels in the neighborhood of these pixels. They then compute the mean difference of classification scores on the whole test dataset each time a perturbation is applied. If the mean difference of scores for a given class is high, then the explanation heat-map is relevant, i.e., perturbing a region assigned a high-importance score changes the output score of the network. For this, they compute the area over perturbation curve metric and show that the LRP method (Sec. 5.1) is the best when compared to gradient backpropagation (Sec. 4.2) and deconvolution (Sec. 4.1) accordingly to the perturbation approach.

Furthermore, methods based on the computation of gradients like the backpropagation, guided backpropagation, and DeconvNet suffer from gradient shattering,[48] where, as the depth of the layers increases the gradients of the loss progressively resembled white noise. This causes the importance values to have high-frequency variations and makes them highly sensitive to small variations in the input. Galli et al.[49] performed adversarial perturbations to their input image using the fast gradient method[50] and DeepFool.[51] They compared the explanation maps of guided Grad-CAM for the image and its perturbed variations using the Dice similarity coefficient after thresholding of importance map with 0.9 as the threshold value. They observed that perturbations in the image strongly affected the saliency maps. The maps of the images before and after perturbations are different, though they note that these differences are not easily perceived by a human viewer. LRP, DeepLIFT, and FEM are not sensitive to this problem as they do not compute gradients.

Adebayo et al.[52] recently proposed two randomization tests to assess the quality of explanation methods. The first one is the model parameter randomization. It consists of comparing the explanations given by a method for a randomly initialized untrained network and a trained network with the same architecture. The model parameters like the weights are randomized for the whole network and also layer by layer. The second test is data randomization, which consists of label randomization: on the contrary to Ref. 47, the input data remain unchanged, but the labels of classes are randomly switched. In both cases, if the explainer is good, the resulting explanation maps will strongly differ from the explanation maps of correctly trained classifiers. As comparison metrics, they have used SSIM, SCC, and PCC of histogram of gradients of explanation maps. The lower the value of the metrics is, the higher is the difference in explanation maps for these tests, and hence better is the explainer. Comparing six white-box methods from the linearization category (see Sec. 4), they concluded that the Grad-CAM and gradient Backpropagation pass this test and are better at explaining the network from the data and parameters that have been learnt.

7.2 *Human Perception-Based Evaluation*

The perturbation-based methods give a way to assess the quality of the explainer without any reference to human perception. A desirable property of every explanation map is that it is non-random and highlights only the relevant regions in the image and no more. Nevertheless, the question arises on what is the target of explanations. In most cases, the target is a human and they are often not AI experts. Therefore, many methods use a qualitative assessment based on human inspection to evaluate and compare different maps. To evaluate the Grad-CAM maps, Selvaraju et al.[30] conducted user surveys to find which maps and models they find reliable. Cruciani et al.[53] demonstrated the usefulness of LRP to visualize relevant features on brain magnetic resonance imaging modality for multiple sclerosis classification. It is observed that the LRP heat maps are sparse and not always as intuitive, which is also illustrated by Fig. 10. It is, therefore, important to consider the user, who will be using the maps while choosing a method to explain a network. For the domain of medical images, the specialist might require a map that provides holistic explanations for them to be able to interpret and trust the network decision. Although human measurements on the quality of these maps are useful, they are time-consuming and could introduce bias and inaccurate evaluations.[54]

Another way to evaluate explainers consists of comparing the explanation maps with human understanding of visual content. The objective way to perform this is by calculating the comparison metrics between explanation maps and gaze fixation density maps (GFDMs) of human experts performing the same visual classification task as the trained network. This can also be done by a comparison of explanation maps and important regions in the images identified by human experts by manual contouring. Mohseni et al.[55] performed a similar experiment. The final goal of explanations is to increase a user's trust in AI systems and especially for image classification tasks in our case. Accordingly, Mohseni et al.[55] observed that providing nonsensical explanations (i.e., those that do not align with users' expectations) may harm the users' reported trust and observed reliance on the system.

Thus in our experiments for an initial evaluation of the methods presented in this paper, we follow the strategy of comparison of explanation maps with user expectations for a given visual classification task. We measure them by GFDMs built upon gaze fixations of observers in a task-driven psychovisual experiment. The usual metrics of comparison of saliency maps, such as PCC, SSIM, and L1 norm of difference (similarity), can also be used here. The higher the value of correlation is, the stronger is the trust of the user in the explainer. Hence in this paper, we compare the presented white-box methods on the datasets with human gaze fixations recorded in a task-driven visual experiment of recognition of architectural styles of Mexican architectural heritage, introduced in Ref. 56 and publicly available in Ref. 57. As comparison metrics, we have chosen PCC and L1 norm of difference (SIM). We have taken a VGG-16 network initialized with ImageNet weights and retrained it on the complete Mexculture dataset from Ref. 58. The results of the comparison of explanation maps from different white-box methods and the GFDM for this network are shown in Table 1. Here the metrics have been computed on 284 images of the MexCulture dataset with available GFDMs for images that have not been used for training.

From this table, one can see that FEM is the most similar to the GFDM in SIM-L1 norm and that Grad-CAM has the highest correlation with the GFDM. The methods belong to the linearization category (Sec. 4). Integrated gradients and SmoothGrad are designed to reduce the noise from gradient backpropagation heatmaps. From this table, it can be seen that the similarity and PCC metrics are improved for these methods when compared to gradient backpropagation. Grad-CAM maps have the highest similarity to the GFDM maps from the linearization category of methods. Figure 11 shows the heatmaps of different white-box methods for a sample image from the dataset. A zero image (black) has been taken as the reference image for the DeepLIFT visualization in this figure. It can be seen that the LRP method highlights the contrasted edges between the structure and sky to be the most relevant regions.

Muddamsetty et al.[59] also created a dataset comprising the user saliency maps in the form of GFDMs of medical experts on retinal images. They compare Grad-CAM and their own explainer maps with GFDM. Metrics such as area under the curve (AUC) and Kullback–Leibler divergence[46] for the two maps were computed for the comparison. They show that the saliency map of both explainers closely aligns with human experts.

Table 1 Comparison of explanation maps of different methods with the GFDM for the VGG16 network trained on the MexCulture dataset.[56]

Method	SIM	PCC
Gradient backpropagation	0.4982 ± 0.051	0.2422 ± 0.142
Guided backpropagation	0.5724 ± 0.063	0.0779 ± 0.067
SmoothGrad	0.5769 ± 0.062	0.1867 ± 0.113
Integrated gradients	0.5747 ± 0.063	0.1390 ± 0.082
Grad-CAM	0.5857 ± 0.104	0.3969 ± 0.290
LRP	0.5728 ± 0.063	0.0754 ± 0.069
DeepLIFT	0.5846 ± 0.063	0.2749 ± 0.081
FEM	0.5999 ± 0.087	0.3558 ± 0.289

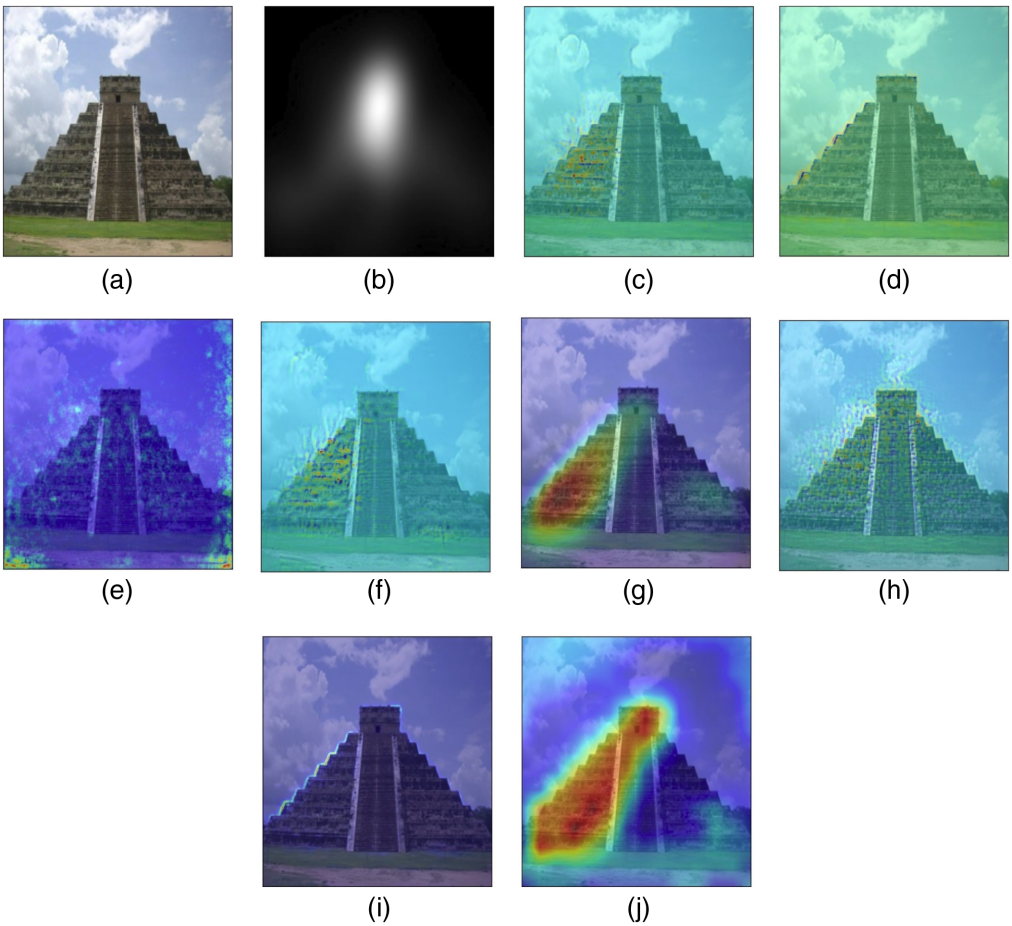

Fig. 11 Heatmap visualizations for the different white-box methods for a sample image from the MexCulture dataset along with the GFDM. (a) Image, (b) GFDM, (c) gradient backpropagation, (d) guided backpropagation, (e) SmoothGrad, (f) integrated gradients, (g) Grad-CAM, (h) DeepLIFT, (i) LRP, and (j) FEM.

When the GFDMs are not available, and this is usually the case in real-world applications, the importance map produced by the explainer can be compared with already existing methods, which have been assessed. Thus for FEM,[37] the authors compare the saliency maps obtained by FEM with gradient-based methods. In their work they show that the FEM maps are most similar, in terms of PCC and L_1 similarity, to the Grad-CAM method.

Methods can also be compared based on their computation time. The choice of methods is limited in their application for a larger set of images or real-time feedback if they have a long computation time. Fuad et al.[37] observed that gradient-based methods including Grad-CAM have longer computation times and FEM was faster in comparison.

8 Conclusion

In this review, we have attempted to provide a comprehensive overview of the current explanation methods for CNNs in image classification tasks. After a short excursion into black-box methods, which explain the decision by masking the input, we focused on white box methods as they can leverage the extra information that is available from the knowledge of the architecture of the network.

The methods that we have discussed are focused on explaining the decision for a single-input image by creating saliency maps that attribute importance scores to each of the pixels based on its contribution to the final output of the CNN. Multiple approaches have been used to calculate this contribution, and based on recent works, we proposed a categorization of methods to understand similar approaches and compare their performance.

The first category we presented is the methods based on linearization of CNNs (Sec. 4) that approximate a CNN as a linear function and explain the decision based on gradient calculation. The deconvolution-based approach and gradient backpropagation were shown to be similar except in the rectification step (ReLU operation), whereas the guided backpropagation method combines the rectification steps of these methods to improve visualization. SmoothGrad and integrated gradients computed explanation maps of gradient backpropagation over multiple variations of the input image and combined them to produce visualizations with reduced noise. On the other hand, the Grad-CAM method computed the gradient only till the last conv layer and used an interpolation method to create the explanation method.

The second category considered were the methods based on network structure (Sec. 5). LRP and DeepLIFT methods define local redistribution rules to redistribute the relevance based on the output score from the last layer to the input. DeepLIFT varies in the regard that uses a reference image and assigns relevance based on the difference in the responses of the neurons for the reference image and the input image. FEM combines the idea of Grad-CAM and uses statistical filtering to identify the strong features from the last conv layer and uses interpolation to create the final explanation map.

The final and third categories are methods based on the adversarial approach (Sec. 6). These methods are distinct from the other categories as they involve the use of generative networks and adversarial attack-based methods to explain the important pixels in the image. Unlike the other categories, there are only a few recent methods that have employed this approach but are interesting to be explored as they give insights about the security aspect of the networks and might be used in real-world applications.

Finally, we have analyzed different ways of evaluation of explanation methods: by perturbation/randomization and by comparing with human perception of images. We share the point that an explainer is good when the explanations match human intuitions contrarily to the perturbation/randomization methods, which assess the explainer only from the classifier's perspective. Thus we have conducted experience on comparison of a bunch of explainers with respect to human GFDMs obtained from a task-driven psychovisual experiment. FEM and Grad-CAM methods were seen to be holistic and more human interpretable methods based on our experiments. Collecting these kinds of maps is often time-consuming and further methods for the evaluation of explanation methods have to be developed.

This review has been proposed from the image understanding perspective. However, the explanation of AI classifiers is necessary for various kinds of imaging and signal data. The

crucial aspect is the bridge between explainability and trustworthiness of AI, and this is a research question open for further contribution from the image research community.

9 Appendix A: Filtering Operation in DeconvNet and Gradient Calculation

Consider a convolution neuron as shown in Fig. 12, where the operation performed is $Y * F = O$, with Y as the input, F the convolution layer filter, and O the output. In a standard CNN during backpropagation of the loss L, the neuron receives a partial gradient of $\partial L/\partial O$ and the gradients to be calculated are $\partial L/\partial Y$ and $\partial L/\partial F$.

According to the chain rule, the calculation of the partial gradient $\partial L/\partial Y$ is given by

$$\frac{\partial L}{\partial Y} = \frac{\partial L}{\partial O} \cdot \frac{\partial O}{\partial Y}. \tag{28}$$

To go through a step-by-step calculation of these gradients we suppose that the input Y is a matrix of 3×3, the filter F is a matrix of size 2×2, and the convolution operation is the one with a stride of 1. Then the corresponding matrices and the loss gradient that is backpropagated from the following layer would be as shown in the following equation:

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}, F = \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}, O = \begin{bmatrix} o_{11} & o_{12} \\ o_{21} & o_{22} \end{bmatrix}, \frac{\partial L}{\partial O} = \begin{bmatrix} \frac{\partial L}{\partial o_{11}} & \frac{\partial L}{\partial o_{12}} \\ \frac{\partial L}{\partial o_{21}} & \frac{\partial L}{\partial o_{22}} \end{bmatrix}. \tag{29}$$

The equations for the calculation of the convolution during the forward pass yields the expressions of the O as shown in the following equation:

$$\begin{aligned} o_{11} &= y_{11}f_{11} + y_{12}f_{12} + y_{21}f_{21} + y_{22}f_{22}, \\ o_{12} &= y_{12}f_{11} + y_{13}f_{12} + y_{22}f_{21} + y_{23}f_{22}, \\ o_{21} &= y_{21}f_{11} + y_{22}f_{12} + y_{31}f_{21} + y_{32}f_{22}, \\ o_{22} &= y_{22}f_{11} + y_{23}f_{12} + y_{32}f_{21} + y_{33}f_{22}. \end{aligned} \tag{30}$$

So to calculate the partial gradient of the loss with respect to the input, the first calculation that needs to be done is $\partial O/\partial Y$. A single calculation of this expression is shown in Eq. (31) based on Eq. (30). The rest of the terms can be calculated similarly:

$$\frac{\partial o_{11}}{\partial y_{11}} = f_{11}, \frac{\partial o_{11}}{\partial y_{12}} = f_{12}, \frac{\partial o_{11}}{\partial y_{21}} = f_{21}, \frac{\partial o_{11}}{\partial y_{22}} = f_{22}. \tag{31}$$

Subsequently, the partial gradient of the loss with respect to the input would be given by following equation:

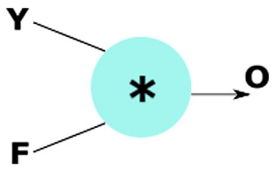

Fig. 12 Convolution operation neuron in a CNN.

$$\frac{\partial L}{\partial y_{11}} = \frac{\partial L}{\partial o_{11}} \cdot f_{11}, \frac{\partial L}{\partial y_{12}} = \frac{\partial L}{\partial o_{11}} \cdot f_{12} + \frac{\partial L}{\partial o_{12}} \cdot f_{11}, \frac{\partial L}{\partial y_{13}} = \frac{\partial L}{\partial o_{12}} \cdot f_{12}$$
$$\frac{\partial L}{\partial y_{21}} = \frac{\partial L}{\partial o_{11}} \cdot f_{21} + \frac{\partial L}{\partial o_{21}} \cdot f_{11}, \frac{\partial L}{\partial y_{22}} = \frac{\partial L}{\partial o_{11}} \cdot f_{22} + \frac{\partial L}{\partial o_{12}} \cdot f_{21} + \frac{\partial L}{\partial o_{21}} \cdot f_{12} + \frac{\partial L}{\partial o_{22}} \cdot f_{11}$$
$$\frac{\partial L}{\partial y_{23}} = \frac{\partial L}{\partial o_{12}} \cdot f_{22} + \frac{\partial L}{\partial o_{22}} \cdot f_{12}, \frac{\partial L}{\partial y_{31}} = \frac{\partial L}{\partial o_{21}} \cdot f_{21}$$
$$\frac{\partial L}{\partial y_{32}} = \frac{\partial L}{\partial o_{21}} \cdot f_{22} + \frac{\partial L}{\partial o_{22}} \cdot f_{21}, \frac{\partial L}{\partial y_{33}} = \frac{\partial L}{\partial o_{22}} \cdot f_{22}. \tag{32}$$

Thus the partial gradient $\partial L/\partial Y$ when calculated using the chain rule can be written as a full convolution (when the loss matrix is zero-padded to have full convolution operation) of the 180-deg inverted filter, i.e., the filter matrix has been flipped vertically and then horizontally as shown in Eq. (33), and the loss gradient matrix $\partial L/\partial O$:

$$F = \begin{bmatrix} f_{22} & f_{21} \\ f_{12} & f_{11} \end{bmatrix}. \tag{33}$$

DeconvNet (Sec. 4.1) uses the same operation at the filtering step. Thus the deconvolution step of the DeconvNet and the calculation of the gradient with respect to the input at a convolution layer are equivalent.

Acknowledgments

This research was supported by the University of Bordeaux/LaBRI. The authors declare that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. C. Garcia and M. Delakis, "Convolutional face finder: a neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(11), 1408–1423 (2004).
2. P. P. de San Roman et al., "Saliency driven object recognition in egocentric videos with deep CNN: toward application in assistance to neuroprostheses," *Comput. Vis. Image Understanding* **164**, 82–91 (2017).
3. Z. Wen et al., "Denoising convolutional neural network inspired via multi-layer convolutional sparse coding," *J. Electron. Imaging* **30**(2), 023007 (2021).
4. P. Martin et al., "Fine grained sport action recognition with twin spatio-temporal convolutional neural networks," *Multimedia Tools Appl.* **79**(27–28), 20429–20447 (2020).
5. J. Shang et al., "Moving object properties-based video saliency detection," *J. Electron. Imaging* **30**(2), 023005 (2021).
6. N. Wu et al., "Deep neural networks improve radiologists' performance in breast cancer screening," *IEEE Trans. Med. Imaging* **39**(4), 1184–1194 (2019).
7. K. Aderghal et al., "Improving Alzheimer's stage categorization with convolutional neural network using transfer learning and different magnetic resonance imaging modalities," *Heliyon* **6**(12), e05652 (2020).
8. M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining*, pp. 1135–1144 (2016).
9. S. Lapuschkin et al., "Unmasking clever Hans predictors and assessing what machines really learn," *Nat. Commun.* **10**(1), 1–8 (2019).
10. M. P. Ayyar et al., "Explaining 3D CNNS for Alzheimer's disease classification on sMRI images with multiple ROIs," in *IEEE Int. Conf. Image Process. (ICIP), Accepted to 2021*, IEEE (2021).

11. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. Learn. Represent. (ICLR)*, pp. 1–14 (2015).
12. W. Samek et al., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Vol. **11700**, Springer Nature, Switzerland (2019).
13. D. Petkovic et al., "Random forest model and sample explainer for non-experts in machine learning: two case studies," *Lect. Notes Comput. Sci.* **12663**, 62–75 (2021).
14. R. Guidotti et al., "A survey of methods for explaining black box models," *ACM Comput. Surveys* **51**(5), 1–42 (2018).
15. O. Boz, "Extracting decision trees from trained neural networks," in *Proc. Eighth ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining,* ACM, pp. 456–461 (2002).
16. N. Frosst and G. E. Hinton, "Distilling a neural network into a soft decision tree," in *Comprehensibility and Explanation in AI and ML (CEX), AI*IA, CEUR Workshop Proc.* (2017).
17. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vision*, Springer, pp. 818–833 (2014).
18. J. Deng et al., "ImageNet: a large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, IEEE, pp. 248–255 (2009).
19. R. Fong and A. Vedaldi, "Explanations for attributing deep neural network predictions," *Lect. Notes Comput. Sci.* **11700**, 149–167 (2019).
20. J. Yosinski et al., "How transferable are features in deep neural networks?" in *Proc. Annu. Conf. Neural Inf. Process. Syst., NeurIPS 2014*, pp. 3320–3328 (2014).
21. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.* **25**, 1097–1105 (2012).
22. I. Goodfellow et al., *Deep Learning*, Vol. **1**, MIT Press, Cambridge (2016).
23. A. Zemmari and J. Benois-Pineau, *Deep Learning in Mining of Visual Content*, Springer, Switzerland (2020).
24. K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: visualising image classification models and saliency maps," in *Proc. 2nd Int. Conf. Learn. Represent., ICLR 2014, Workshop Track Proc.* (2014).
25. J. T. Springenberg et al., "Striving for simplicity: the all convolutional net," in *Proc.3rd Int. Conf. Learn. Represent., ICLR 2015, Workshop Track Proc.* (2015).
26. A. M. Obeso et al., "Image annotation for Mexican buildings database," *Proc. SPIE* **9970**, 99700Y (2016).
27. A. Montoya Obeso et al., "Organizing cultural heritage with deep features," in *Proc. 1st Workshop Struct. and Understanding of Multimedia Heritage Contents*, pp. 55–59 (2019).
28. M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. Int. Conf. Mach. Learn., PMLR*, pp. 3319–3328 (2017).
29. D. Smilkov et al., "SmoothGrad: removing noise by adding noise," arXiv:1706.03825, 1–10 (2017).
30. R. R. Selvaraju et al., "Grad-CAM: visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vision*, pp. 618–626 (2017).
31. Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1798–1828 (2013).
32. B. Zhou et al., "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 2921–2929 (2016).
33. S. Bach et al., "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One* **10**(7), e0130140 (2015).
34. G. Montavon et al., "Layer-wise relevance propagation: an overview," *Lect. Notes Comput. Sci.* **11700**, 193–209 (2019).
35. G. Montavon et al., "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognit.* **65**, 211–222 (2017).
36. A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. Int. Conf. Mach. Learn., PMLR*, pp. 3145–3153 (2017).

37. K. A. A. Fuad et al., "Features understanding in 3D CNNS for actions recognition in video," in *Proc. Tenth Int. Conf. Image Process. Theory, Tools and Appl. (IPTA)*, IEEE, pp. 1–6 (2020).
38. J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 7132–7141 (2018).
39. C. Szegedy et al., "Intriguing properties of neural networks," in *Int. Conf. Learn. Represent. (ICLR) (Poster)*, pp. 1–10 (2014).
40. G. Tao et al., "Attacks meet interpretability: attribute-steered detection of adversarial samples," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, pp. 7728–7739 (2018).
41. A. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *Proc. AAAI Conf. Artif. Intell.*, Vol. **32(1)** (2018).
42. A. Ignatiev, N. Narodytska, and J. Marques-Silva, "On relating explanations and adversarial examples," in *Proc. Adv. Neural Inf. Process. Syst., NeurIPS 2019*, pp. 15857–15867 (2019).
43. J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.* **23**(5), 828–841 (2019).
44. M. Charachon et al., "Combining similarity and adversarial learning to generate visual explanation: application to medical image classification," arXiv:2012.07332, To be published in ICPR 2020 (2020).
45. I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst., NeurIPS 2014*, pp. 2672–2680 (2014).
46. Z. Bylinskii, et al., "What do different evaluation metrics tell us about saliency models?" *IEEE Trans. Pattern Anal. Mach. Intell.* **41**(3), 740–757 (2018).
47. W. Samek et al., "Evaluating the visualization of what a deep neural network has learned," *IEEE Trans. Neural Networks Learn. Syst.* **28**(11), 2660–2673 (2017).
48. D. Balduzzi et al., "The shattered gradients problem: if resnets are the answer, then what is the question?" in *Proc. Int. Conf. Mach. Learn., PMLR*, pp. 342–350 (2017).
49. A. Galli et al., "Reliability of explainable artificial intelligence in adversarial perturbation scenarios," *Lect. Notes Comput. Sci.* **12663**, 243–256 (2021).
50. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Representat., ICLR* (2015).
51. S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 2574–2582 (2016).
52. J. Adebayo et al., "Sanity checks for saliency maps," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst., NIPS'18*, Curran Associates Inc., Red Hook, New York, pp. 9525–9536 (2018).
53. F. Cruciani et al., "Explainable 3D-CNN for multiple sclerosis patients stratification," *Lect. Notes Comput. Sci.* **12663**, 103–114 (2021).
54. Z. Buçinca et al., "Proxy tasks and subjective measures can be misleading in evaluating explainable AI systems," in *Proc. 25th Int. Conf. Intell. User Interfaces*, pp. 454–464 (2020).
55. S. Mohseni, J. E. Block, and E. D. Ragan, "Quantitative evaluation of machine learning explanations: a human-grounded benchmark," in *26th Inter. Conf. Intell. User Interfaces*, ACM, pp. 22–31 (2021).
56. A. M. Obeso et al., "Comparative study of visual saliency maps in the problem of classification of architectural images with deep CNNS," in *Eighth Inter. Conf. Image Process. Theory, Tools and Appl.*, IEEE, pp. 1–6 (2018).
57. S. G. Vázquez and A. R. R. Acosta, https://api.nakala.fr/data/11280%2F5712e468/1e412e0a43b5635365293b249feb9d53d74b5dc8 (2018).
58. A. M. Obeso et al., "Saliency-based selection of visual content for deep convolutional neural networks: application to architectural style classification," *Multimedia Tools Appl.* **78**(8), 9553–9576 (2019).
59. S. M. Muddamsetty, M. N. S. Jahromi, and T. B. Moeslund, "Expert level evaluations for explainable AI (XAI) methods in the medical domain," *Lect. Notes Comput. Sci.* **12663**, 35–46 (2021).

Meghna P. Ayyar received her MS degree in image processing and computer vision and is currently working as an associate junior researcher at LaBRI, Bordeaux, France. Her topics of interest include computer vision, explainable AI, and applications of deep learning to healthcare.

Jenny Benois-Pineau is a full professor of computer science at the University Bordeaux. Her topics of interest include image/multimedia and artificial intelligence (AI). She has authored and co-authored more than 220 papers and is an associate editor of SPIC, ACM MTAP, senior associate editor JEI SPIE journals. She has Knight of Academic Palms grade and has served in program committees of major international conferences in her field. She is a member of IEEE IVMSP TC committee.

Akka Zemmari is a full professor of computer science at the University of Bordeaux. His research interests include AI, deep learning, distributed algorithms and systems, graphs, randomized algorithms, and security. He is a co-coordinator of AI research axis in LaBRI and has been a researcher in several national, European, and Europe-India research projects. He is the author and co-author of more than 80 research papers, conference proceedings, and book chapters.