CAM-Based Methods Can See Through Walls

Magamed Taimeskhanov, Ronan Sicre, Damien Garreau

CAM-Based Methods Can See through Walls

Magamed Taimeskhanov[1] (✉), Ronan Sicre[2], and Damien Garreau[3]

[1] Université Côte d'Azur, Laboratoire J.A. Dieudonné, CNRS, Nice, France
magamed.taimeskhanov@etu.univ-cotedazur.fr
[2] Centrale Méditerranée, Aix-Marseille Univ., CNRS, LIS, Marseille, France
ronan.sicre@lis-lab.fr
[3] Julius-Maximilians Universität Würzburg, Germany
damien.garreau@uni-wuerzburg.de

Abstract. CAM-based methods are widely-used post-hoc interpretability method that produce a saliency map to explain the decision of an image classification model. The saliency map highlights the important areas of the image relevant to the prediction. In this paper, we show that most of these methods can incorrectly attribute an important score to parts of the image that the model cannot see. We show that this phenomenon occurs both theoretically and experimentally. On the theory side, we analyze the behavior of GradCAM on a simple masked CNN model at initialization. Experimentally, we train a VGG-like model constrained to not use the lower part of the image and nevertheless observe positive scores in the unseen part of the image. This behavior is evaluated quantitatively on two new datasets. We believe that this is problematic, potentially leading to mis-interpretation of the model's behavior.

Keywords: Interpretability · Computer Vision · Convolutional Neural Networks · Class Activation Maps.

1 Introduction

The recent advances of machine learning pervade all applications, including the most critical. However, deep learning models intrinsically possess many parameters, have complicated architectures, and rely on many non-linear operations, preventing the users to get a good grasp of the rationale behind particular decisions. These models are often called "black boxes" for these reasons [1]. In this respect, there is a growing need for interpretability of the models that are used, which gave birth to the field of eXplainable AI (XAI). When the model to explain is already trained, our main topic of interest, this is often called post-hoc interpretability [2, 3, 4].

In the specific case of image classification, the explanations provided to the user often take the form of a saliency map superimposed to the original image, for instance simply looking at the gradient with respect to the input of the network [5]. The message is simple: the areas highlighted by the saliency maps are used by the network for the prediction. When the first layers of the network are

convolutional layers [6], one can take advantage of this and look at the activations of the filters corresponding to the class prediction that we are trying to explain. Indeed, these first layers act like a bank of filters on the input image, and the degree to which they are activated gives us information on the behavior of the network. Thus the first layers possess a certain degree of interpretability, even though it can be challenging to aggregate the information coming from different filters. In any case, the next layers generally consist in a fully-connected neural network, thus suffering from the same caveats as other models. In addition, this second part of the network is equally important for the prediction, but is not taken into account in the explanations we provide if we simply look at activation values.

To solve this problem, a natural idea is to weight each activation map depending on how the second part of the network uses it. In the case of a single additional layer, this is called *class activation maps* [CAM, 7]. The methodology was quickly generalized by [8], using the *average gradient* values of the subsequent layers instead, giving rise to GradCAM, arguably one of the most popular posthoc interpretability method for CNNs. Many extensions are proposed in the following years, we list them in Appendix A and refer to [9] for a recent survey. Without being too technical, for all these methods, the explanations provided consist in a weighted average of the activation maps.

A close inspection of each of these methods reveals that the coefficient associated to each individual map is global, in the sense that the same coefficient is applied to the whole map. The main message of this paper is that this can be problematic, since different parts of the activation map may be used differently by the subsequent layers. Worse, **some parts may even be unused by the subsequent network and still highlighted in the final explanation** (see Figure 1). Thus we believe that, while giving apparently more-than-satisfying results in practice, CAM-based methods should be used with caution, keeping in mind that some parts of the image may be highlighted whereas they are not even seen by the network.

1.1 Related work

This paper is inspired by a line of recent works concerned with the reliability of saliency maps claiming that solely relying on the visual explanation provided by a saliency map can be misleading [10, 11]. [11] introduces a method for altering the input data with imperceptible perturbations which do not change the predicted label, yet generating different saliency maps. On the other hand, [10] shows that numerous saliency methods generate incorrect scores for the input features when the model prediction is invariant to translation of the input data by a constant. It is important to note that neither of these studies specifically challenges the reliability of CAM-based methods.

This perspective on saliency maps is supported by the work of [12], which introduces a randomization-based sanity check indicating that some existing saliency

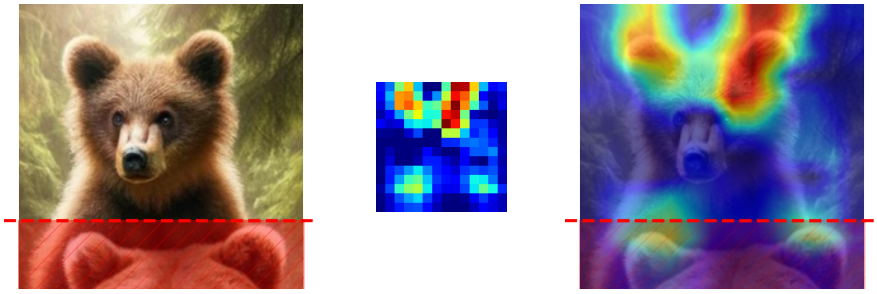

Fig. 1: Example of GradCAM failure on a VGG-like model trained on the ImageNet dataset (masked [**VGG**], see Figure 4). *Left:* original image; *Middle:* GradCAM explanation before up-sampling; *Right:* original image with GradCAM explanation overlayed as a heatmap. The network does not have access to the red part of the image, **but GradCAM does highlight some pixels in this area.**

methods are independent of both the model and the data. We note that GradCAM passes the sanity checks proposed by [12]. [13], proposing HiResCAM, are less positive regarding GradCAM pointing out, as we do, that the use of a global coefficient can produce positive explanations where there should not be. Compared to our work, they provide few theoretical explanations and perform experiments on model which are not using parts of the input image. Posthoc interpretability methods in the image realm (not specific to CNN architectures) have been investigated by other works such as [14] which looked into LIME for images [15].

Taking another angle, [16] directly attacks the reliability of GradCAM saliency maps by adversarial model manipulation, *i.e.*, fine-tuning a model with the purpose of making GradCAM saliency maps unreliable. This is achieved by using a specific loss function tailored to this effect. Our approach is different, as we simply force a strong form of sparsity in the model's parameters, not targeting a specific interpretability method.

1.2 Organization of the paper

We start by looking at GradCAM in Section 2. For a given simple CNN architecture described in Section 2.1, we derive closed-form expressions for its explanations in Section 2.2. Leveraging these expressions, we prove in Section 2.3 that GradCAM explanations are positive at initialization, even though a large part of the weights are set to zero.

In Section 3, we demonstrate experimentally that this phenomenon remains true after training. To this extent, we proceed in two steps. First, we train to a reasonable accuracy a VGG-like model on ImageNet [17] **which does not see the lower part of input images**, described in Section 3.1. Then, we create two

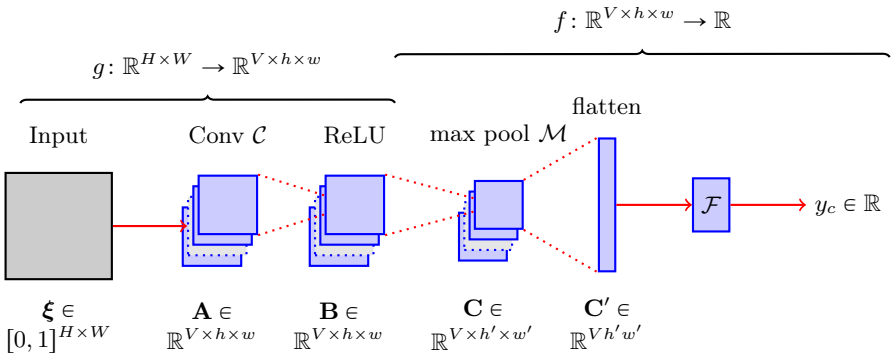

Fig. 2: The model used for the derivation of feature importance scores, [**CNN**]. The number of filters in the convolutional layer $\mathcal{C}$ is $V \in \mathbb{N}^\star$. The size of the max pooling filters $k' \in \mathbb{N}^\star$ is implicitly defined such that $(h', w') = \frac{1}{k'}(h, w)$ in $\mathbb{N}^\star$. The fully-connected neural network $\mathcal{F}(\cdot)$ takes $\mathbf{C}'$ as input and processes it through L layers with ReLU activation functions to produce a raw score y_c, without converting this score into a "probability."

datasets (Section 3.2) consisting in superposition of images of the same class. We show experimentally in Section 3.3 that **CAM-based methods applied to this model wrongly highlights a large portion of the lower part of the images**, misleading the user by showing that the lower part is used for the prediction whereas, by construction it is not. The code for training our model as well as the datasets are provided as supplementary material. Additionally, the code for all experiments is available online.[4] We conclude in Section 4.

2 Mathematical description

The model used for the theoretical analysis done in Section 2.3 is described in Section 2.1, the derivation of GradCAM coefficients in Section 2.2.

2.1 A simple CNN

Let us describe mathematically the model we consider, denoted by [**CNN**] and depicted in Figure 2. On a high-level, [**CNN**] is a $(L+1)$-layers network, consisting in a single convolution / max pooling layer, followed by a L-layers fully-connected neural network with ReLU activations. Thus the case $L = 1$ corresponds to a single convolutional / max pooling layer followed by a linear transformation.

[4] `https://github.com/MagamedT/cam-can-see-through-walls`

More precisely, we consider a grayscale image $\boldsymbol{\xi} \in [0,1]^{H \times W}$ as input. For instance, if we consider the MNIST dataset [18], $(H, W) = (28, 28)$. We note that our analysis can be easily extended to RBG images. The convolutional layer $\mathcal{C}(\cdot)$ consists of V filters $\mathbf{F} = (\mathbf{F}_1, \ldots, \mathbf{F}_V)$, represented as a collection of V matrices of shape $k \times k$.

Formally, the output of the convolution step, $\mathbf{A} := \mathcal{C}(\boldsymbol{\xi}) \in \mathbb{R}^{V \times h \times w}$, is given by:

$$\forall v \in [V],\ \forall (i,j) \in [h] \times [w], \qquad \mathbf{A}^{(v)}_{i,j} = \sum_{p,q=1}^{k} \boldsymbol{\xi}_{i+p-1,j+q-1} \mathbf{F}^{(v)}_{p,q}\,, \tag{1}$$

where $(h, w) = (H - k + 1, W - k + 1)$.

In practice, the filter weights are initialized randomly, typically i.i.d. uniform or Gaussian with proper scaling. There are two main trends on how to scale the variance, either Glorot [19] (also called Xavier), or He [20]. The later with uniform distribution is default for the CNN layer used in PyTorch. However, we assume from now on i.i.d. Gaussian $\mathcal{N}\left(0, \tau^2\right)$ initialization in our analysis for mathematical convenience.

After the convolution step, we apply a ReLU non-linearity, denoted by $\sigma := \max(0, \cdot)$. We define the *rectified activation maps* $\mathbf{B} := \sigma(\mathbf{A}) \in \mathbb{R}^{V \times h \times w}$, where σ is applied coordinate-wise. Next, we consider a down-sampling layer, here a $(k' \times k')$ max pooling $\mathcal{M}(\cdot)$. One can see that the output of the max pooling, $\mathbf{C} := \mathcal{M}(\mathbf{B}) \in \mathbb{R}^{V \times h' \times w'}$, is given by:

$$\forall v \in [V],\ \forall (i',j') \in [h'] \times [w'],\ \mathbf{C}^{(v)}_{i',j'} = \max\left(\mathbf{B}^{(v)}_{k'(i'-1)+1:k'i',k'(j'-1)+1:k'j'}\right)\,, \tag{2}$$

where $(h', w') = \frac{1}{k'}(h, w)$. Note that we assume k' to divide h and w for simplicity.

Finally, let us describe recursively the fully-connected part of [**CNN**], denoted by $\mathcal{F}(\cdot)$:

$$\begin{aligned} \mathcal{F}\colon \mathbb{R}^{Vh'w'} &\to \mathbb{R} \\ \mathbf{C}' &\mapsto h_L\left(\mathbf{C}'\right) \end{aligned} \quad \text{with} \quad \begin{cases} h_0(\mathbf{x}) = \mathbf{x}, \\ h_\ell(\mathbf{x}) = \mathbf{W}^{(\ell)}\, \sigma\left(h_{\ell-1}(\mathbf{x})\right) & \text{for } \ell \in [L], \end{cases} \tag{3}$$

where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is a weight matrix connecting layer $(\ell - 1)$ and ℓ with $\ell \in [L]$ and d_ℓ the size of layer ℓ. Note that we set $d_0 = Vh'w'$, and $d_L = 1$, since we see the output of our model as the un-normalized logit associated to a given class of a prediction problem. We also, denote by $\mathbf{a}^{(\ell)} := h_\ell(\mathbf{C}')$ the non-rectified activation of layer ℓ and $\mathbf{r}^{(\ell)} := \sigma\left(\mathbf{a}^{(\ell)}\right)$ its rectified counterpart.

Summary. The model we consider can be described concisely as $\mathcal{F}(\mathcal{M}(\sigma(\mathcal{C}(\cdot))))$. As explained in introduction, given the nature of CAM-based explanations, **it is convenient to split [CNN] in two functions f and g** for the computations of the next Section 2.2. More precisely, we write

$$\begin{aligned} [\mathbf{CNN}]\colon [0,1]^{H \times W} &\to \mathbb{R} \\ \boldsymbol{\xi} &\mapsto f \circ g(\boldsymbol{\xi}) \end{aligned} \quad \text{with} \quad \begin{cases} g(\boldsymbol{\xi}) := \sigma(\mathcal{C}(\boldsymbol{\xi})) \\ f(\mathbf{C}) := \mathcal{F}(\mathcal{M}(\mathbf{C}))\ . \end{cases} \tag{4}$$

Recall that we refer to Figure 2 for an illustration.

2.2 Closed-form expression

The original idea of CAM [7] was limited to computing the saliency map as a linear combination of the feature maps in the last convolutional layer when $L = 1$. Later, GradCAM [8] removed the architecture constraints by computing the average gradient of each feature map with respect to $\mathbf{B}$. In our notation, we have:

Definition 1 (GradCAM). *For an input $\boldsymbol{\xi}$ and model* [**CNN**]*, the GradCAM feature scores are given by*

$$[\mathbf{GC}] := \sigma\left(\sum_{v=1}^{V} \alpha_v \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,,$$

where each $\alpha_v := \mathrm{GAP}\left(\nabla_{\mathbf{B}^{(v)}} f(\mathbf{B})\right) \in \mathbb{R}$. Here, GAP denotes the global average pooling*, that is, the average of all values, and σ the ReLU as before.*

Definition 1 is of course to be taken coordinate-wise. We note that, in practice, [**GC**] is up-sampled and normalized to produce a saliency map with the *same shape* as the input image. To be more precise, what we define as [**GC**] is the middle panel of Figure 1, whereas the final user will nearly always visualize the right panel. The most important thing to notice in Definition 1 is that α is a **global** coefficient.

We now show why this can be an issue. Looking at Definition 1, whenever the underlying model is not too complicated, one can actually hope to derive a closed-form expression for the feature importance scores of [**GC**] as a function of the model's parameters. This is achieved by:

Proposition 1 (α coefficients for GradCAM, $V = 1$). *Recall that the $\mathbf{a}$ vectors denote the non-rectified activation and $\mathbf{W}$ the weights of the linear part of* [**CNN**]*. Then, for input $\boldsymbol{\xi}$, the* [**GC**] *coefficient α is given by*

$$\alpha = \frac{1}{hw}\sum_{i,j=1}^{h',w'} \sum_{i_1,\ldots,i_{L-1}=1}^{d_1,\ldots,d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_1}^{(1)},\ldots,\mathbf{a}_{i_{L-1}}^{(L-1)}>0} \prod_{p=1}^{L} (\mathbf{W}_{i_p,i_{p-1}}^{(p)})^\top\,,$$

where we set $i_0 := (i,j)$ and $i_L = 1$.

From Proposition 1, we immediately deduce a closed-form expression for GradCAM explanations. We note that Proposition 1 can be readily extended to an arbitrary number of filters $V > 1$, in which case the $\mathbf{a}$ and $\mathbf{W}$ should be interpreted as corresponding to the relevant $v \in [V]$.

The proof of Proposition 1 can be found in Appendix C. In Appendix A, we describe mathematically several other CAM-based methods in the setting of

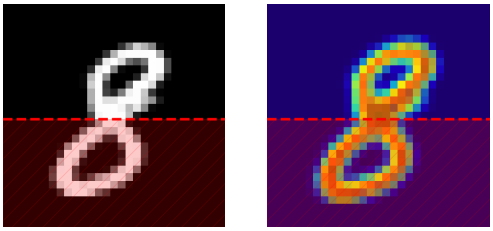

Fig. 3: Illustration of Theorem 1 on an MNIST [18] digit (*left panel*). We set to zero the lower part of $\mathbf{W}$ for [**CNN**], initialize the filter values and remaining weights to i.i.d. $\mathcal{N}(0,1)$, and run GradCAM to get a saliency map (*right panel*). Even though our network does not see the red part of the image, **GradCAM does highlight some pixels in this area**, as predicted by Theorem 1.

[**CNN**]: XGradCAM [21], GradCAM++ [22], HiResCAM [13], ScoreCAM [23] and AblationCAM [24]. A close inspection of these definitions reveals that they also use global weighting coefficients applied to the corresponding activation maps, with the notable exception of HiResCAM.

2.3 Theoretical analysis

Leveraging the results of Section 2.2, we are able to describe precisely the behavior of GradCAM at initialization for [**CNN**], specifically when the classifier part of our model comprises a single layer ($L = 1$). This analysis is justified by existing works [25, 26, 27, 28, 29] showing that, in certain regimes, neural networks stay "near initialization" during training. As announced, we conduct this analysis when the network does not have access to the lower part of the image. Our main result is:

Theorem 1 (Expected GradCAM scores, $L = 1$, masked [CNN]). *Let $\boldsymbol{\xi} \in [0,1]^{H \times W}$ be an input image. Let $\mathbf{m} := \boldsymbol{\xi}_{i:i+k-1,j:j+k-1}$ be the patch of $\boldsymbol{\xi}$ corresponding to index $(i,j) \in [h] \times [w]$. Assume that h' is even, and $\mathbf{W}_{:,-\frac{h'}{2}:,:} = 0$. Assume that the filter values and the non-zero weights are initialized i.i.d. $\mathcal{N}\left(0, \tau^2\right)$. Then, if the number of filters V is greater than 20, we have the following expected lower bound on the GradCAM explanation for pixel (i,j):*

$$\mathbb{E}\left[[\mathbf{GC}]_{i,j}\right] = \mathbb{E}\left[\sigma\left(\sum_{v=1}^{V} \alpha_v \mathbf{B}_{i,j}^{(v)}\right)\right] \geqslant \frac{V-20}{\sqrt{V}} \sqrt{\frac{h'w'}{16\pi}} \frac{\tau^2}{hw} \|\mathbf{m}\|_2 \,, \tag{5}$$

where the expectation in the previous inequality is taken with respect to initialization of the filters and the remaining weights of the linear layer.

Setting $\mathbf{W}_{:,-\frac{h'}{2}:,:}$ to 0 disables the weights within $\mathbf{W}$ that are connected to the lower half part of the activation map $\mathbf{C}_{:,-\frac{h'}{2}:,:}$, effectively preventing [**CNN**] from

accessing the lower half of $\mathbf{C}$. In turn, [**CNN**] does not see the lower half of ξ, up to side effects. The main consequence of Theorem 1 is that, when the number of filters associated to the class to explain is large enough, $[\mathbf{GC}]_{i,j}$ is positive in expectation if some pixels are activated in the receptive field associated to (i, j). Thus **GradCAM highlights all parts of the image where there is some "activity," even though this information is not used by the network in the end.** We illustrate Theorem 1 in Figure 3. The main limitation of this analysis is its focus on the behavior at initialization: we investigate in the following whether this behavior also happens after training. Another limitation is the restriction to a single linear layer, but we note that taking $L = 1$ in the fully connected part of [**CNN**] is a dominant approach since ResNet [30].

The proof of Theorem 1 can be found in Appendix D. The key ingredient of the proof is obtaining a probabilistic control of $\sum_q \alpha_q \mathbf{B}^{(q)}$. We note that a similar analysis is possible for other expressions of α, thus other CAM-based methods.

3 Experiments

We know ask the following question: are the consequences of Theorem 1 true after training, and for a more realistic model? To this extent, we train a CNN-based model which by construction cannot access some specified part of the input which we call the *dead zone* (see Figure 4, details in Section 3.1). Clearly, since the dead zone does not influence the output, it should not contain positive model explanations. To test whether this is true, we create two datasets (Section 3.2). Each item of the first one is composed of two images from ImageNet with the same label in both the seen and the unseen part of the image. The second dataset is built using generative models on the same categories with two objects in each image located in the seen and unseen part as well. We then check whether CAM-based methods wrongly highlight areas in the dead zone in Section 3.3.

3.1 Model

Model definition. The CNN used in our experiments is a modification of a classical VGG16 architecture [31] which we call [**VGG**]. Whereas the original VGG16 model is composed of 5 convolutional blocks including either 2 or 3 convolutional layers with ReLU and max pooling, followed by 3 dense layers. In [**VGG**] we remove the last max pooling (in the fifth convolutional block) and we further apply a mask on selected neurons of the first dense layer so the layer can not see the lower part of the activation maps, see Figure 4 for more details.

Masking. We forbid the network from seeing the dead zone in a very simple way: in the first dense layer $\mathbf{W}$, which has size $4096 \times (256 \times 14 \times 14)$, we permanently set to 0 a band of height 9 corresponding to the lower weights. Formally, this means setting $\mathbf{W}_{:,:,-9:,:} = 0$, which is denoted in red above $\mathbf{W}$ in Figure 4. Effectively, we are building a wall that stops all information flowing from the last convolutional layer to the remainder of the network. Since the weights $\mathbf{W}$

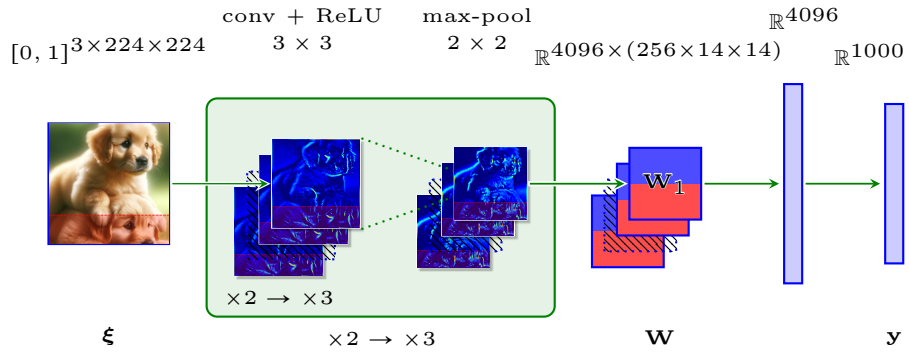

Fig. 4: Our masked VGG16-based model trained on ImageNet with 87.0% top-5 accuracy. The down weights $\mathbf{W}_{:,:,-9:,:}$ are set to 0 and not updated during training. Only the up weights $\mathbf{W}_{:,:,:5,:}$ and the other parameters undergo training. This setting implies that every red part in the channels does not impact the prediction scores, meaning that they are not used. Symbol $\times2 \rightarrow \times3$ means the model first uses the green block twice, with each time having 2 consecutive convolutions. Then, it uses the green block three times, with each time having 3 consecutive convolutions. There is no max pooling after the last convolution.

are directly connected to the final activation map $\mathbf{B} \in \mathbb{R}_+^{256\times14\times14}$, this masking effectively zeroes out the lower sections in each channel denoted by $\mathbf{B}_{:,-9:,:}$. We can trace back the zeroed activations in $\mathbf{B}$ to the preceding activation map $\mathbf{C}$, pinpointing the exact patches in $\mathbf{C}$ that correspond (after convolution) to the features observed in the zeroed activation of $\mathbf{B}$. Because of the side effects in the computation of convolutions, this area of $\mathbf{C}$ is slightly smaller: some pixel activation will still play a role in the model's prediction. Repeating this process until we reach the original image yields a dead zone of height 54 pixels, highlighted in red above $\boldsymbol{\xi}$ in Figure 4, which covers 24% of the image area. As we mentioned earlier, the other main difference with VGG16 is the removing of the final max pooling layer. This leads to a larger activation layer, allowing us to set weights to zero without hindering too much the network's ability, see Figure 5. We note that [**VGG**] bears a strong resemblance to [**CNN**]. The main difference is that the convolutional layer of [**CNN**] is replaced by several convolutional blocks in [**VGG**], see Figure 4.

Training. We train [**VGG**] on Imagenet-1k [17] using classical data augmentation recipe, *i.e.*, random flip and random crop. As optimization algorithm, we use stochastic gradient descent with momentum, weight decay, and a learning rate scheduler. To observe the slight accuracy drop induced by masking a significant part of images, we train a baseline model without masking. We report the train loss and the validation accuracy across training in Figure 5.

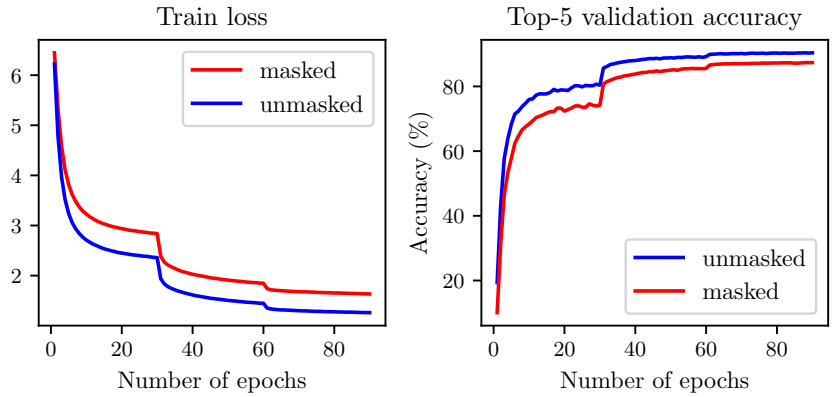

Fig. 5: Plots of the training loss and the top-5 validation accuracy of our unmasked and masked VGG16-like models on Imagenet-1k (2012) [17]. The unmasked [**VGG**] (baseline) and our masked [**VGG**] yield 87.0%, resp. 90.4% top-5 accuracy and 66.5% resp. 71.5% top-1 accuracy on the validation set.

Comparison to SOTA. We also compare the validation top-1 and top-5 accuracy of the VGG16 model found in the PyTorch repository. Our [**VGG**] without max pooling and no masking offers the same performance: 71.5% top-1 and 90.4% top-5 accuracy on the validation set. As we mention in Figure 5, our model [**VGG**] with masking has lower performance, which is expected as a fourth of the input image, $\boldsymbol{\xi}_{:,171{:}224,:}$, is unseen by the model. We obtain 66.5%, resp. 71.5%, top-1 and 87.0%, resp. 90.4%, top-5 accuracy on the validation set for our masked [**VGG**], resp. unmasked [**VGG**]. Nevertheless, we see that [**VGG**] is a **realistic network able to predict ImageNet classes with reasonable accuracy**. We believe that the drop in accuracy is only minor because ImageNet images are centered, and there is enough information in the upper part of the image to achieve near-perfect prediction.

3.2 Proposed datasets

Objective. To assess how much CAM-based saliency maps emphasize irrelevant areas of an image, we introduce two new datasets in which we control the positions of the image elements using two techniques: cutmix [32] and generative model. More precisely, we produce two datasets, called STACK-MIX and STACK-GEN. Where each image contains two objects, one in the bottom part of the image which is the dead zone for [**VGG**], and the second subject at the top of the image. Therefore, the subject at the center of the image will be mainly responsible for the top-1 predicted score by our masked [**VGG**].

STACK-MIX. We first generate labels for our datasets by ramdomly sampling 100 classes from the 398 first labels of Imagenet, which corresponds to animals.

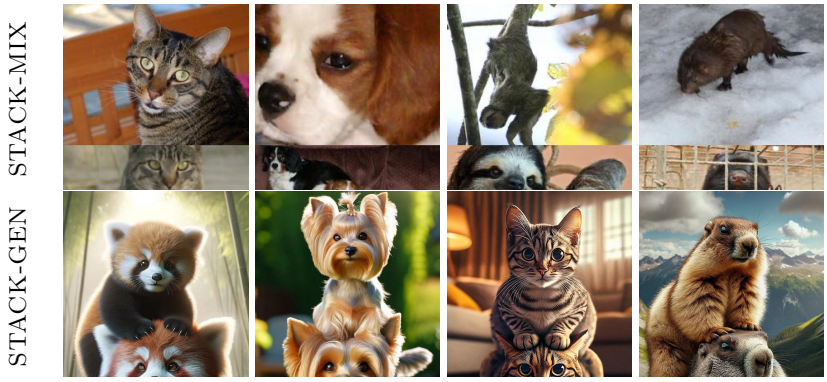

Fig. 6: Sampled images from both of our datasets, *i.e.*, STACK-MIX and STACK-GEN.

The first dataset, called STACK-MIX, consists of 100 images featuring one image from each of the 100 classes. Each example $\boldsymbol{\xi}$ is created by mixing, in a cutmix [32] fashion, two images $(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2)$ with same label and sampled randomly in the **validation set** of Imagenet as follows:

$$\boldsymbol{\xi} := \begin{pmatrix} (\boldsymbol{\xi}_1)_{:,\,:170,\,:} \\ (\boldsymbol{\xi}_2)_{:,\,171:224,\,:} \end{pmatrix} \in [0,1]^{3\times 224\times 224}\,, \tag{6}$$

meaning that we create a composite image $\boldsymbol{\xi}$ by superposing an upper vertical slice, taken from the top region of $\boldsymbol{\xi}_1$ with size $3 \times 170 \times 224$, with a lower vertical slice, taken from the bottom region of $\boldsymbol{\xi}_2$ with size $3 \times 54 \times 224$. Finally, the quality of the generated images is verified through manual inspection. This dataset lacks realism due to the distinct separation between the two subjects. We address this issue with the help of generative models.

STACK-GEN. The second dataset, called STACK-GEN, consists of 100 images featuring one image from each of the same 100 classes. It was generated using ChatGPT + DALL·E 3 [33, 34] by sampling prompts of the following form: "A photo of {animal name} stacked on top of {same animal name}". The word "stacked" determines the positions of the subjects in the generated image, which proceeds as follows: first, ChatGPT refines the original prompt to enhance its suitability for DALL·E 3, then the image is generated. We then preprocess the generated images by selectively editing them to minimize the background and centering the focus on the two animals. This editing involves cropping the images to a 1:1 ratio, ensuring one animal is predominantly within the dead zone as defined by our [**VGG**], while the other is positioned in the upper part of the new image. Figure 6 shows examples of the created images. Note that both datasets are provided in the supplementary material and online.[5]

[5] `https://github.com/MagamedT/cam-can-see-through-walls`

Table 1: Activity in the unseen part of the image, measured by $\mu(\cdot) \times 100$ for several CAM-based methods on both proposed datasets (only images in the validation set are considered).

methods	STACK-MIX ↓	STACK-GEN ↓
GradCAM [8]	22.7 ± 13.4	21.6 ± 11.6
GradCAM++ [22]	28.8 ± 8.1	28.5 ± 7.9
XGradCAM [21]	23.8 ± 9.0	22.8 ± 9.0
ScoreCAM [23]	19.9 ± 10.3	18.5 ± 10.6
Opti-CAM [9]	32.7 ± 7.9	32.0 ± 7.8
AblationCAM [24]	21.0 ± 9.9	20.8 ± 9.6
EigenCAM [35]	51.7 ± 19.7	55.8 ± 21.6
HiResCAM [13]	0.0 ± 0.0	0.0 ± 0.0

3.3 Results

For our **[VGG]**, we generate saliency maps from various CAM-based methods on our two datasets, STACK-MIX and STACK-GEN, using the predicted category for each example. We used publicly available implementations whenever possible. Regarding Opti-CAM, since our model differs from the one described by [9], we have adjusted the learning rate and number of epochs of the optimization step to achieve a low average drop, as described in the original paper. For each method, we measure how much of the CAM-based saliency maps emphasize the unseen part, *i.e.*, the dead zone. We use the metric $\mu(\cdot)$ defined for a upscaled saliency map $\mathbf{\Lambda} \in \mathbb{R}_+^{224\times 224}$ as follows:

$$\mu(\mathbf{\Lambda}) := \frac{\|\mathbf{\Lambda}_{171:224,:}\|_2}{\|\mathbf{\Lambda}\|_2}\,, \tag{7}$$

where $\|\cdot\|_2$ is the ℓ^2-norm and the lower part of the image $\boldsymbol{\xi}_{:,\,171:224,:}$ is unseen by our **[VGG]**. We note that for a saliency map $\mathbf{\Lambda}$, the lower $\mu(\mathbf{\Lambda})$, the better.

The results can be found in Table 1, Figure 7 and 8. We observe that every CAM-based methods, except HiResCAM, highlights unseen parts of an image to some extent. Moreover, the observation are consistent over both datasets.

We believe that HiResCAM avoids this problem because of how its weighting coefficients are computed. Following the notation of Section 2.1, these can be written $\alpha_v^{(4)} := \nabla_{\mathbf{B}^{(v)}} f(\mathbf{B}) \in \mathbb{R}^{14\times 14}$, and are applied *globally* to $\mathbf{B}^{(v)}$ (see Definition 4 in the Appendix for more details). Because of the masking used in our model, the lower part of $\alpha_v^{(4)}$ is zeroed out, and therefore HiResCAM does not show activity in the lower part of the image.

We notice that, while at first glance HiResCAM appears to perform well in our setting, it has another issue: since $(\alpha_v^{(4)})_{-9:,:} = 0$, the upscaled HiResCAM's

saliency map $\mathbf{\Lambda} \in \mathbb{R}_+^{224\times224}$ will be zero out in a larger area than the deadzone. Namely, $\mathbf{\Lambda}_{89:224,:} = 0$ which represents 61% of the input image area compared to the 24% of the deadzone. This issue can be observed in Figure 7.

4 Conclusion

In this paper, we looked into several CAM-based methods, with a particular focus on GradCAM. We showed that they can highlight parts of the input image that are provably not used by the network. This was also showed theoretically, looking at the behavior of GradCAM for a simple, masked CNN at initialization: the saliency map is positive in expectation, even in areas which are unseen by the network. Experimentally, this phenomenon appears to remain true, even on a realistic network trained to a good accuracy on ImageNet.

As future work, we would like to extend the theory to a ResNet-like architecture and other CAM-based methods, such as LayerCAM [36]. We also would like to multiply the number of images in our two new datasets, with the hope that this framework can become a standard check for saliency maps explanations.

Acknowledgements

This work was funded in part by the French Agence Nationale de la Recherche (grant number ANR-19-CE23-0009-01 and ANR-21-CE23-0005-01). Most of this work was realized while DG was employed at Université Côte d'Azur. We thank Jenny Benois-Pineau for her valuable insights.

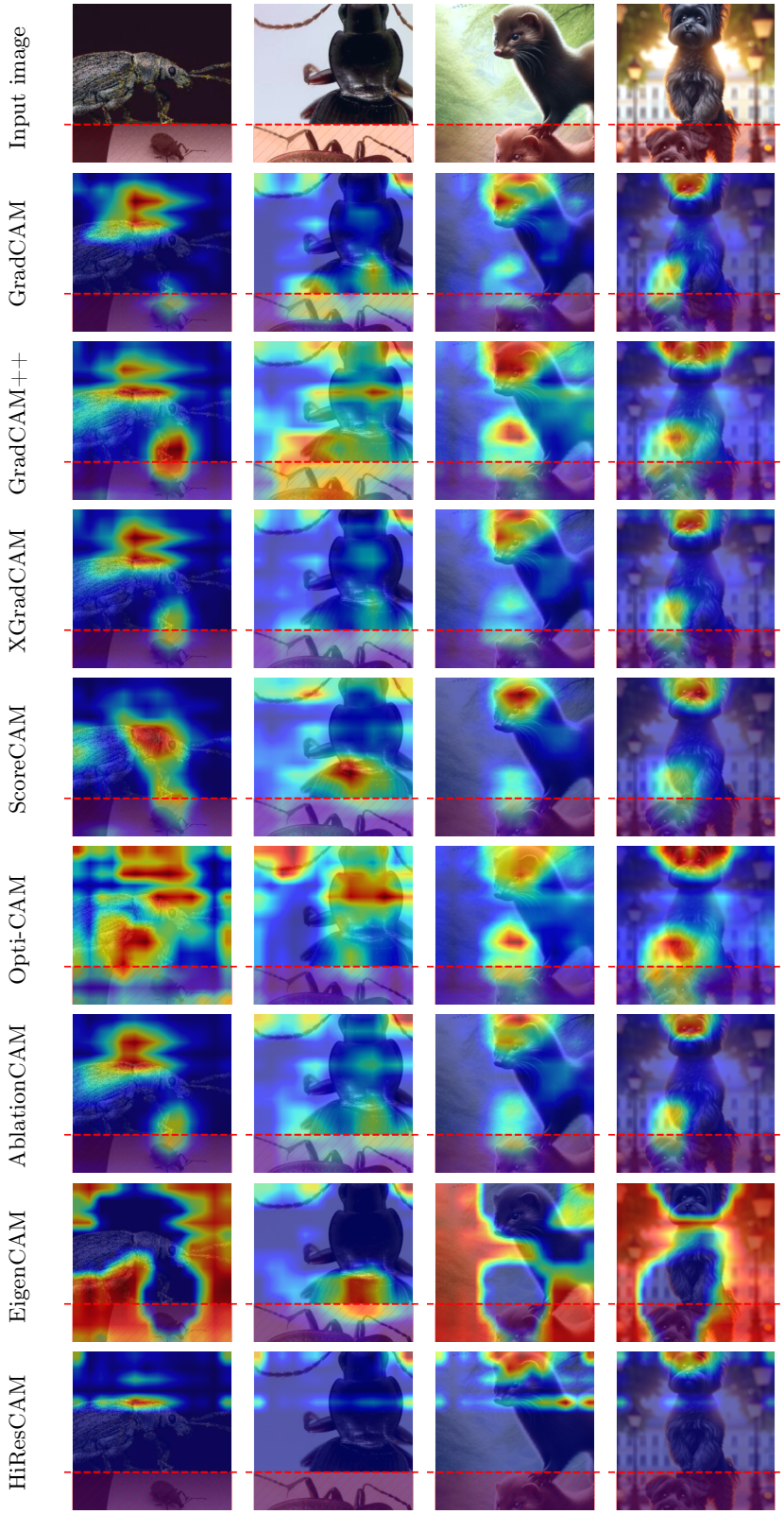

Fig. 7: Saliency maps given by the considered CAM-based methods for [**VGG**]. With the notable exception of HiResCAM, all method highlight parts of images from STACK-GEN and STACK-MIX which are unseen by the network. The lower part in red is unseen by the model.

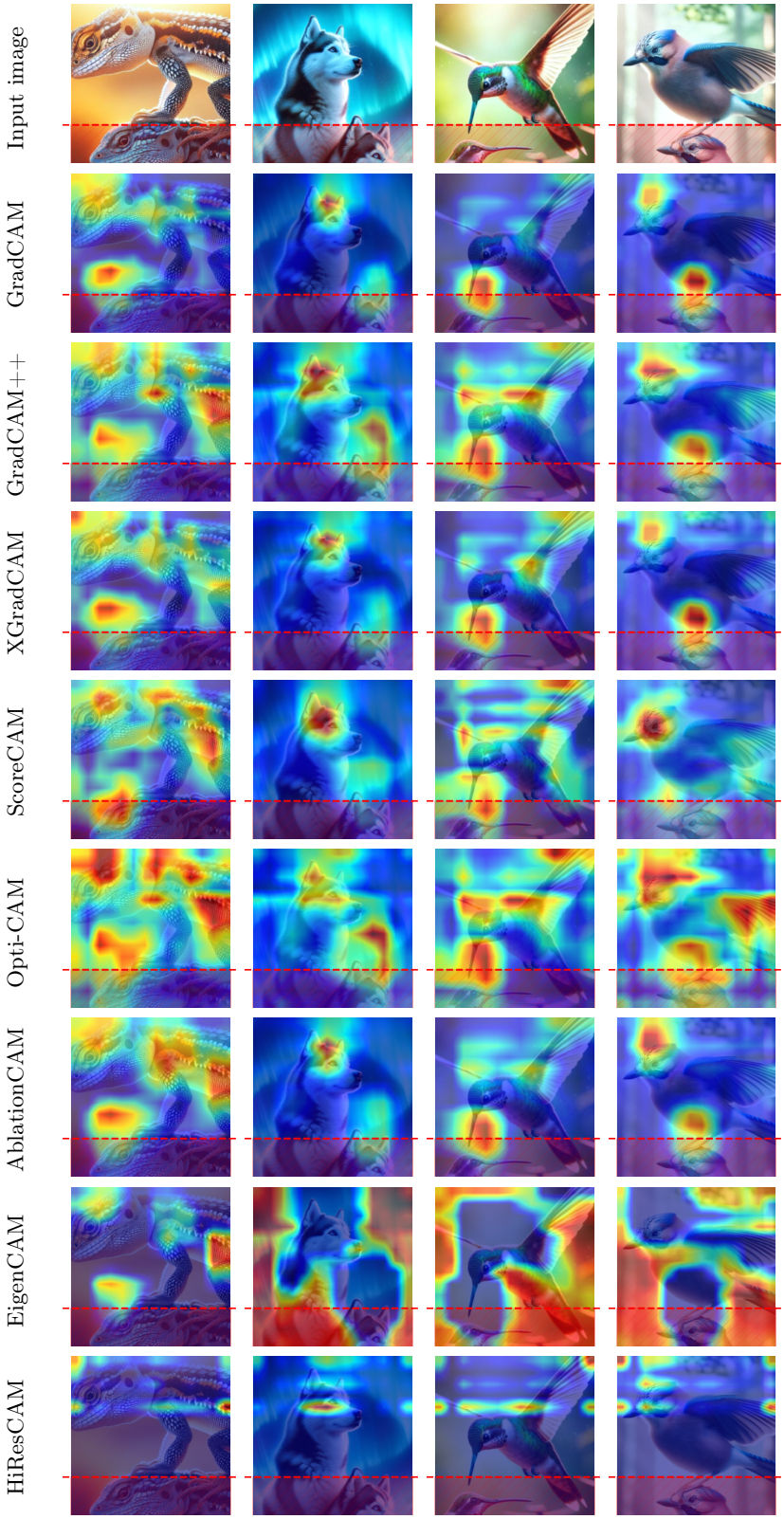

Fig. 8: Saliency maps given by the considered CAM-based methods for [**VGG**]. With the notable exception of HiresCAM, they all highlight parts of images from STACK-GEN which are unseen by the network (this is denoted by the red, rectangular shape in the lower part of the image).

Bibliography

[1] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks*, 1997.

[2] Zachary C. Lipton. The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery. 2018.

[3] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A Survey on Neural Network Interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.

[4] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 2021.

[5] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv preprint arXiv:1312.6034*, 2013.

[6] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980.

[7] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[8] 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. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[9] Hanwei Zhang, Felipe Torres, Ronan Sicre, Yannis Avrithis, and Stephane Ayache. Opti-CAM: Optimizing saliency maps for interpretability. *arXiv preprint arXiv:2301.07002*, 2023.

[10] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (Un)reliability of saliency methods. *Explainable AI: Interpreting, explaining and visualizing deep learning*, 2019.

[11] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of Neural Networks is Fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[12] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems*, 2018.

[13] Rachel Lea Draelos and Lawrence Carin. Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks. *arxiv preprint 2011.08891*, 2021.

[14] Damien Garreau and Dina Mardaoui. What does LIME really see in images? In *International Conference on Machine Learning*. PMLR, 2021.

[15] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2016.

[16] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling Neural Network Interpretations via Adversarial Model Manipulation. In *Advances in Neural Information Processing Systems*, 2019.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[21] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based Grad-CAM: Towards Accurate Visualization and Explanation of CNNs. In *31st British Machine Vision Conference*, 2020.

[22] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks. In *IEEE Winter Conference on Applications of Computer Vision*, 2018.

[23] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu. Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[24] Saurabh Desai and Harish G. Ramaswamy. Ablation-CAM: Visual Explanations for Deep Convolutional Network via Gradient-free Localization. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[25] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. *Advances in Neural Information Processing Systems*, 2019.

[26] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient Descent Finds Global Minima of Deep Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[27] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A Convergence Theory for Deep Learning via Over-Parameterization. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[28] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the Convergence Rate of Training Recurrent Neural Networks. *Advances in Neural Information Processing Systems*, 2019.

[29] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep relu networks. *Machine Learning*, 2020.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[31] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[32] S. Yun, D. Han, S. Chun, S. Oh, Y. Yoo, and J. Choe. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[33] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 2020.

[34] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-Shot Text-to-Image Generation. In *International Conference on Machine Learning*, 2021.

[35] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-CAM: Visual Explanations for Deep Convolutional Neural Networks. *SN Computer Science*, 2021.

[36] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. LayerCAM: Exploring Hierarchical Class Activation Maps for Localization. *IEEE Transactions on Image Processing*, 2021.

[37] Maxime Beauchamp. On numerical computation for the distribution of the convolution of N independent rectified Gaussian variables. *Journal de la Société Française de Statistique*, 2018.

A Other definitions

The work of [21] proposed an alternative to [**GC**] by re-scaling the gradient in the weighting coefficient α. The main objective of this alternative is to obtain a method that satisfies two axioms (sensitivity and conservation properties) defined by [21], which, according to their findings, enhances visualization performance w.r.t. specific metrics.

Definition 2 (XGradCAM). *In our notation, for an input* $\boldsymbol{\xi}$ *and model* [**CNN**]*, the XGradCAM feature scores are given by*

$$[\mathbf{XC}] := \sigma\left(\sum_{v=1}^{V} \alpha_v^{(2)} \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,,$$

where each $\alpha_v^{(2)} := \mathrm{GAP}\left(\frac{\mathbf{B}^{(v)}}{\left\|\mathbf{B}^{(v)}\right\|_1} \odot \nabla_{\mathbf{B}^{(v)}} f(\mathbf{B})\right)$ *is the* global average pooling *of the gradient of* f *at* $\mathbf{B}^{(v)}$ *rescaled by the normalized activation* $\frac{\mathbf{B}^{(v)}}{\left\|\mathbf{B}^{(v)}\right\|_1}$.

GradCAM++ [22] then modified GradCAM by computing rectified gradient together with second and third order derivative information. This alternative appears to produce better saliency maps in cases where the input image contains multiple subjects.

Definition 3 (GradCAM++). *In our notation, for an input* $\boldsymbol{\xi}$ *and model* [**CNN**]*, the GradCAM++ feature scores are given by*

$$[\mathbf{C}+] := \sigma\left(\sum_{v=1}^{V} \alpha_v^{(3)} \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,,$$

where each $\alpha_v^{(3)} := \mathrm{GAP}\left(\frac{\partial^2_{\mathbf{B}^{(v)}} f(\mathbf{B})}{2\partial^2_{\mathbf{B}^{(v)}} f(\mathbf{B}) + \left\|\mathbf{B}^{(v)}\right\|_1 \partial^3_{\mathbf{B}^{(v)}} f(\mathbf{B})} \odot \sigma\left(\nabla_{\mathbf{B}^{(v)}} f(\mathbf{B})\right)\right)$ *is the* global average pooling *of the gradient of* f *at* $\mathbf{B}^{(v)}$ *rescaled by second and third order derivatives defined as*

$$\partial^n_{\mathbf{B}^{(v)}} f(\mathbf{B}) := \left(\partial^n_{\mathbf{B}^{(v)}_{i,j}} f(\mathbf{B})\right)_{i,j\in[h]\times[w]}\,.$$

More recently, HiResCAM [13] proposed to replace the averaging of the gradient over the map by the *element-wise* multiplication between gradient and activation. In our notation:

Definition 4 (HiResCAM). *For an input* $\boldsymbol{\xi}$ *and model* [**CNN**]*, the HiResCAM feature scores are given by*

$$[\mathbf{HC}] := \sigma\left(\sum_{v=1}^{V} \alpha_v^{(4)} \odot \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,,$$

where $\alpha_v^{(4)} := \nabla_{\mathbf{B}^{(v)}} f(\mathbf{B}) \in \mathbb{R}^{h\times w}$ *and* $\odot$ *is the element-wise matrix product.*

The work of [23] got rid of the dependency on gradients of the weighting coefficients α.

Definition 5 (ScoreCAM). *In our notation, for an input* $\boldsymbol{\xi}$ *and model* $[\mathbf{CNN}]$*, the ScoreCAM feature scores are given by*

$$[\mathbf{SC}] := \sigma\left(\sum_{v=1}^{V} \alpha_v^{(5)} \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,.$$

In the previous display, $\alpha^{(5)} = \mathrm{softmax}(\beta^{(5)})$*, where* $\beta_v^{(5)} := [\mathbf{CNN}]\left(\boldsymbol{\xi} \odot H^{(v)}\right) - [\mathbf{CNN}]\left(\boldsymbol{\xi}_b\right)$*,* $H^{(v)} := s\left(\mathrm{Up}\left(\mathbf{B}^{(v)}\right)\right)$ *the upsampled and normalized activation map* $\mathbf{B}^{(v)}$*,* $s(\cdot)$ *a normalization function that maps matrix values into* $[0,1]$*, and* $\mathrm{Up}(\cdot)$ *an upsampling function which resize a matrix to the size of* $\boldsymbol{\xi}$*. The baseline input image* $\boldsymbol{\xi}_b$ *can be taken as* $\boldsymbol{\xi}_b := \boldsymbol{\xi}$*.*

Similar to ScoreCAM, the work of [9] introduces Opti-CAM, a method that uses optimization to compute the weighting coefficient α.

Definition 6 (Opti-CAM). *Define*

$$H_\beta := s\left(\mathrm{Up}\left(\sum_{v=1}^{V} \mathrm{softmax}(\beta)_v \mathbf{B}^{(v)}\right)\right)\,,$$

where $s(\cdot)$ *is a normalization function that maps matrix values into* $[0,1]$*, and* $\mathrm{Up}(\cdot)$ *an upsampling function which resizes a matrix to the size of* $\boldsymbol{\xi}$*. Take* $\alpha^{(6)} = \mathrm{softmax}(\beta^\star)$*, where* $\beta^\star$ *is solution to the optimization problem*

$$\underset{\beta\in\mathbb{R}^V}{\mathrm{Maximize}}\, [\mathbf{CNN}]\left(\boldsymbol{\xi} \odot H_\beta\right)\,.$$

Then, for an input $\boldsymbol{\xi}$*, the Opti-CAM feature scores are given by*

$$[\mathbf{OC}] := \sigma\left(\sum_{v=1}^{V} \alpha_v^{(6)} \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,.$$

It is evident from the definitions of ScoreCAM and OptiCAM that the ReLU function $\sigma(\cdot)$ is redundant when $\mathbf{B}$ represents the rectified activation maps, since they are already positive.

Finally, AblationCAM [24] removes each activation to observe the impact on the prediction. Ablations resulting in larger drops receive higher weights. AblationCAM is similar to ScoreCAM: a sort of masking is performed to observe changes in prediction and no gradient is required. However, both methods require lots of forward passes through the network.

Definition 7 (AblationCAM). *In our notation, for an input* $\boldsymbol{\xi}$ *and model* [**CNN**]*, the AblationCAM feature scores are given by*

$$[\mathbf{AC}] := \sigma\left(\sum_{v=1}^{V} \alpha_v^{(7)} \mathbf{B}^{(v)}\right) \in \mathbb{R}_+^{h\times w}\,,$$

where each $\alpha_v^{(7)} := \frac{\mathbf{y}-\mathbf{y}_v}{\mathbf{y}}$*,* $\mathbf{y} := [\mathbf{CNN}](\boldsymbol{\xi})$ *the predicted score by our model and* $\mathbf{y}_v$ *is the output of* [**CNN**] *when the activation map* $\mathbf{B}^{(v)}$ *is zero out.*

It should be clear that the method we use to compute the weighting coefficients α of [**GC**], in Proposition 1, yields the coefficients $\alpha^{(4)}$ for [**HC**] and $\alpha^{(2)}$ for [**XC**], since in proving Proposition 1 we have computed $\nabla_{\mathbf{B}^{(v)}} f(\mathbf{B})$, which is the challenging part.

B Technical results

As a consequence of Eq. (1), $\mathbf{A}$ (seens as a vector of size $h \times w$) is a centered Gaussian vector, with covariance matrix given by

$$\forall (i,j),(i',j') \in ([h]\times[w])^2,\ \left(\Sigma^A\right)_{(i,j),(i',j')} = \mathrm{Tr}\left(\xi_{i:i+k,j:j+k}^{\top}\xi_{i':i'+k,j':j'+k}\right)\,. \tag{8}$$

Because of this simple remark, we can describe precisely the distribution of $\mathbf{A}$ (and, further, $\mathbf{B}$), thanks to the following lemmas. We denote by ϕ the density of the standard Gaussian distribution and Φ its cumulative distribution function.

Lemma 1 (Expectation of rectified Gaussian). *Let* $X \sim \mathcal{N}\left(\mu, \tau^2\right)$*, we get the following expectation for the rectified Gaussian* $X^+ := \sigma(X)$*:*

$$\mathbb{E}\left[X^+\right] = \mu\Phi\left(\frac{\mu}{\tau}\right) + \tau\phi\left(\frac{-\mu}{\tau}\right)\,.$$

Proof. See [37]. □

Lemma 2 (Law of convolution). *Let* $\mathbf{m} \in \mathbb{R}^{k\times k}$ *and* $\mathbf{F} \sim \mathcal{N}\left(0, \tau^2 \mathrm{Id}_k\right)$*, then the convolution* $\mathbf{F} \star \mathbf{m}$ *has distribution* $\mathcal{N}\left(0, \left(\tau \left\|\mathbf{m}\right\|_2\right)^2\right)$*.*

Proof. First let us remind that:

$$\mathbf{F} \star \mathbf{m} = \sum_{i,j=1}^{k} \mathbf{F}_{i,j}\mathbf{m}_{i,j}\,.$$

This lemma is derived from rapid calculations that proceed as follows, utilizing the fact that the elements $\left(\mathbf{F}_{i,j}\right)_{i,j}$ are independent and identically distributed

(i.i.d.):

$$\mathbb{E}\left[\mathbf{F} \star \mathbf{m}\right] = 0\,,$$
$$\mathrm{Var}\left(\mathbf{F} \star \mathbf{m}\right) = \tau^2 \sum_{i,j=1}^{k} \mathbf{m}_{i,j}^2 = \tau^2 \left\|\mathbf{m}\right\|_2^2 .$$

□

Straightforward computations yields:

Lemma 3 (Moments of squared rectified Gaussian). *Let $X \sim \mathcal{N}\left(0, \tau^2\right)$ of density $f(\cdot)$, we get the following two moments for the squared rectified Gaussian $(X^+)^2$:*

$$\mathbb{E}\left[(X^+)^2\right] = \frac{\tau^2}{2} \quad \textit{and} \quad \mathrm{Var}\left((X^+)^2\right) = \frac{5}{4}\tau^4\,.$$

C Proof of Proposition 1

Notation. Before starting the proof, let us recall some notation. In Section 2.1, we defined $\mathbf{a}^{(u)} := h_u\left(\mathbf{C}'\right)$ the non-rectified activation of layer u, $\mathbf{r}^{(u)} := \sigma\left(\mathbf{a}^{(u)}\right)$ its rectified counterpart. We also defined $\mathbf{C}$ the max pooled rectified activation $\mathcal{M}(\mathbf{B})$. In this proof, we write f_u the function that maps the input of the u-th layer $\mathbf{r}^{(u-1)}$ to its output $\mathbf{r}^{(u)}$ as follows $f_u(\mathbf{r}^{(u-1)}) := \sigma\left(\mathbf{W}^{(u)}\mathbf{r}^{(u-1)}\right)$ with $u \in [L]$.

Now let us turn to the computation of $\nabla_{\mathbf{B}} f(\mathbf{B}) \in \mathbb{R}^{(h\times w)\times 1}$ with $\mathbf{B} \in \mathbb{R}^{h\times w}$. By the chain rule, we get $\nabla_{\mathbf{B}} f(\mathbf{B}) = \nabla_{\mathbf{B}}\left(\mathcal{F}\circ\mathcal{M}\right)(\mathbf{B}) = \nabla_{\mathbf{B}}\mathcal{M}\cdot\nabla_{\mathcal{M}(\mathbf{B})}\mathcal{F}$ where:

$$\begin{aligned}\nabla_{\mathbf{C}}\mathcal{F} &= \nabla_{\mathbf{C}} f_1 \cdot \nabla_{f_1(\mathbf{C})}\left(\mathbf{W}^{(L)} f_{L-1}\circ\cdots\circ f_2\right)\\ &= \left(\nabla_{\mathbf{C}} f_1\right)\cdot\left(\nabla_{\mathbf{r}^{(1)}} f_2\right)\cdots\left(\nabla_{\mathbf{r}^{(L-2)}} f_{L-1}\right)\cdot\left(\mathbf{W}^{(L)}\right)\,.\end{aligned}$$

We compute $\nabla_{\mathbf{r}^{(i)}} f_{i+1} \in \mathbb{R}^{d_i\times d_{i+1}}$ as follows:

$$\nabla_{\mathbf{r}^{(i)}} f_{i+1} = \nabla_{\mathbf{r}^{(i)}}\,\sigma\left(\mathbf{W}^{(i+1)}\mathbf{r}^{(i)}\right) = \left(\mathbf{W}^{(i+1)}\right)^{\top}\nabla_{\mathbf{a}^{(i+1)}}\,\sigma\left(\mathbf{a}^{(i+1)}\right)\,.$$

Now let us remark that, for all $x \in \mathbb{R}^d$, $\partial_i\,\sigma\left(x\right)_j = \partial_i\,\sigma\left(x_j\right) = \mathbb{1}_{x_j>0}\mathbb{1}_{i=j}$, since σ is applied element-wise. Thus

$$\nabla_{\mathbf{a}^{(i+1)}}\,\sigma\left(\mathbf{a}^{(i+1)}\right) = \begin{pmatrix} \mathbb{1}_{\mathbf{a}_1^{(i+1)}>0} & 0 & \cdots & 0\\ 0 & \mathbb{1}_{\mathbf{a}_2^{(i+1)}>0} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \mathbb{1}_{\mathbf{a}_{d_{i+1}}^{(i+1)}>0}\end{pmatrix}.$$

Finally,

$$\begin{aligned}\nabla_{\mathbf{r}^{(i)}} f_{i+1} &= \left(\mathbf{W}_{1,:}^{(i+1)^\top} \cdots \mathbf{W}_{d_{i+1},:}^{(i+1)^\top}\right) \nabla_{\mathbf{a}^{(i+1)}} \sigma\left(\mathbf{a}^{(i+1)}\right) \\ &= \left(\mathbb{1}_{\mathbf{a}_1^{(i+1)}>0} \mathbf{W}_{1,:}^{(i+1)^\top} \cdots \mathbb{1}_{\mathbf{a}_{d_{i+1}}^{(i+1)}>0} \mathbf{W}_{d_{i+1},:}^{(i+1)^\top}\right).\end{aligned}$$

Now to compute $\nabla_{\mathbf{C}} \mathcal{F}$ let us start from the end with $\nabla_{\mathbf{r}^{(L-2)}} f_{L-1} \mathbf{W}^{(L)^\top}$:

$$\begin{aligned}\nabla_{\mathbf{r}^{(L-2)}} f_{L-1} \mathbf{W}^{(L)^\top} &= \left(\mathbb{1}_{\mathbf{a}_1^{(L-1)}>0} \mathbf{W}_{1,:}^{(L-1)^\top} \cdots \mathbb{1}_{\mathbf{a}_{d_{L-1}}^{(L-1)}>0} \mathbf{W}_{d_{L-1},:}^{(L-1)^\top}\right) \mathbf{W}^{(L)^\top} \\ &= \left(\sum_{i_{L-1}=1}^{d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_{L-1}}^{(L-1)}>0} \mathbf{W}_{i_{L-1},:}^{(L-1)^\top} \mathbf{W}_{1,i_{L-1}}^{(L)}\right).\end{aligned}$$

Similarly,

$$\nabla_{\mathbf{r}^{(L-3)}} f_{L-2} = \left(\mathbb{1}_{\mathbf{a}_1^{(L-2)}>0} \mathbf{W}_{1,:}^{(L-2)^\top} \cdots \mathbb{1}_{\mathbf{a}_{d_{L-2}}^{(L-2)}>0} \mathbf{W}_{d_{L-2},:}^{(L-2)^\top}\right),$$

so the next computation gives

$$\begin{aligned}&\nabla_{\mathbf{r}^{(L-3)}} f_{L-2} \nabla_{\mathbf{r}^{(L-2)}} f_{L-1} \mathbf{W}^{(L)^\top} \\ &= \nabla_{\mathbf{r}^{(L-3)}} f_{L-2} \left(\sum_{i_{L-1}=1}^{d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_{L-1}}^{(L-1)}>0} \mathbf{W}_{i_{L-1},:}^{(L-1)^\top} \mathbf{W}_{1,i_{L-1}}^{(L)}\right) \\ &= \nabla_{\mathbf{r}^{(L-3)}} f_{L-2} \begin{pmatrix} \sum_{i_{L-1}=1}^{d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_{L-1}}^{(L-1)}>0} \mathbf{W}_{i_{L-1},1}^{(L-1)^\top} \mathbf{W}_{1,i_{L-1}}^{(L)} \\ \vdots \\ \sum_{i_{L-1}=1}^{d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_{L-1}}^{(L-1)}>0} \mathbf{W}_{i_{L-1},d_{L-2}}^{(L-1)^\top} \mathbf{W}_{1,i_{L-1}}^{(L)} \end{pmatrix} \\ &= \left(\sum_{i_{L-2},i_{L-1}=1}^{d_{L-2},d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_{L-2}}^{(L-2)}, \mathbf{a}_{i_{L-1}}^{(L-1)}>0} \mathbf{W}_{i_{L-2},:}^{(L-2)^\top} \mathbf{W}_{i_{L-1},i_{L-2}}^{(L-1)^\top} \mathbf{W}_{1,i_{L-1}}^{(L)}\right).\end{aligned}$$

Now we can conclude by straightforward induction:

$$\begin{aligned}&\nabla_{\mathbf{C}} \mathcal{F} \\ &= \left(\sum_{i_1,\ldots,i_{L-1}=1}^{d_1,\ldots,d_{L-1}} \mathbb{1}_{\mathbf{a}_{i_1}^{(1)},\ldots,\mathbf{a}_{i_{L-1}}^{(L-1)}>0} (\mathbf{W}_{i_1,:}^{(1)})^\top (\mathbf{W}_{i_2,i_1}^{(2)})^\top \ldots (\mathbf{W}_{i_{L-1},i_{L-2}}^{(L-1)})^\top \mathbf{W}_{1,i_{L-1}}^{(L)}\right).\end{aligned}$$

Now to finish we need to compute $\nabla_{\mathbf{B}} \mathcal{M}(\mathbf{B})$, to do so, we suppose that the convolutional layer $\mathcal{C}$ returns an reshaped rectified activation map $\mathbf{B}$ such that we get the following gradient:

$$\begin{pmatrix} \mathbf{D}_1 & 0 & 0 \\ 0 & \mathbf{D}_2 & 0 \\ \vdots & & \vdots \\ 0 & \cdots & \mathbf{D}_{h' \times w'} \end{pmatrix} \in \mathbb{R}^{((k')^2 \times h' \times w') \times (h' \times w')} = \mathbb{R}^{(h \times w) \times (h' \times w')}$$

with $\mathbf{D}_i \in \mathbb{R}^{(k')^2 \times 1}$ being the i-th block of indicator functions defined as follows:

$$\mathbf{D}_i := \begin{pmatrix} \mathbb{1}_{\max \mathbf{m}_i = \mathbf{m}_{i,(1,1)}} \\ \mathbb{1}_{\max \mathbf{m}_i = \mathbf{m}_{i,(1,2)}} \\ \vdots \\ \mathbb{1}_{\max \mathbf{m}_i = \mathbf{m}_{i,(2,1)}} \\ \vdots \\ \mathbb{1}_{\max \mathbf{m}_i = \mathbf{m}_{i,(k',k')}} \end{pmatrix},$$

where $\mathbf{m}_{i,(p,q)} := (\mathbf{m}_i)_{p,q}$ and $\mathbf{m}_i := \mathbf{B}_{k'(i'-1)+1:k'i',k'(j'-1)+1:k'j'} \in \mathbb{R}_+^{k' \times k'}$, $(i',j') := ((i-1)/k'+1, (i-1 \mod k')+1)$ is the i-th patch of $\mathbf{B}$ (the patch are ordered from left to right, starting from the top of $\mathbf{B}$). Also note that $\cdot/\cdot$ is the integer division.

Finally, we can compute $\nabla_{\mathbf{B}} f(\mathbf{B})$ as follows:

$$\begin{aligned} \nabla_{\mathbf{B}} f(\mathbf{B}) &= \nabla_{\mathbf{B}} \mathcal{M} \cdot \nabla_{\mathcal{M}(\mathbf{B})} \mathcal{F} \\ &= \begin{pmatrix} \mathbf{D}_1 \rho_1 \\ \vdots \\ \mathbf{D}_{h' \times w'} \rho_{h' \times w'} \end{pmatrix} \\ &\in \mathbb{R}^{(h \times w) \times 1} \end{aligned} \tag{9}$$

where $\rho_b \in \mathbb{R}$, $b \in [h' \times w']$ is defined as follows:

$$\rho_b := \sum_{i_1,\ldots,i_{L-1}=1}^{d_1,\ldots,d_{L-1}} \mathbb{1}_{\mathbf{a}^{(1)}_{i_1},\ldots,\mathbf{a}^{(L-1)}_{i_{L-1}} > 0} (\mathbf{W}^{(1)}_{i_1,b})^\top (\mathbf{W}^{(2)}_{i_2,i_1})^\top \ldots (\mathbf{W}^{(L-1)}_{i_{L-1},i_{L-2}})^\top \mathbf{W}^{(L)}_{1,i_{L-1}} .$$

To compute α, we simply take the average of the components of the previous display, $\nabla_{\mathbf{B}} f(\mathbf{B})$, which yields

$$\begin{aligned} \alpha &= \frac{1}{hw} \sum_{k=1}^{(k')^2} \sum_{j=1}^{h' \times w'} \mathbf{D}_{j,k} \rho_j \\ &= \frac{1}{hw} \sum_{j=1}^{h' \times w'} \left(\sum_{k=1}^{(k')^2} \mathbf{D}_{j,k} \right) \rho_j \\ &= \frac{1}{hw} \sum_{j=1}^{h' \times w'} \rho_j \,, \end{aligned}$$

since $\sum_{k=1}^{(k')^2} \mathbf{D}_{j,k} = 1$ with fixed j (it is also the case in practice when using automatic differentiation) and $\mathbf{D}_{j,k} := (\mathbf{D}_j)_k$. □

D Proof of Theorem 1

Before jumping into the proof, let us remark that the left-hand side of Eq. (5) is non-negative, thus there is nothing to prove if $\|\mathbf{m}\|_2 = 0$. Thus we will assume that $\|\mathbf{m}\|_2 > 0$ from now on.

Separating the randomness. There are two sources of randomness: the weights of the filters $\mathbf{F} = (\mathbf{F}^{(1)}, \dots, \mathbf{F}^{(V)})$ and the coefficients of the linear layer $\mathbf{W}$. We start this proof by dissociating these two sources of randomness. More precisely, following Definition 1, and using the law of total expectation, we compute the expected GradCAM heatmap at coordinates (i, j) as

$$\mathbb{E}\left[\sigma\left(\sum_{q=1}^{V} \alpha_q \mathbf{B}_{i,j}^{(q)}\right)\right] = \mathbb{E}\left[\mathbb{E}\left[\sigma\left(\sum_{q=1}^{V} \alpha_q \mathbf{B}_{i,j}^{(q)}\right)\middle|\mathbf{F}\right]\right]. \tag{10}$$

Using the computed GradCAM coefficient in Proposition 1 with $L = 1$, we get $\alpha_q = \frac{1}{hw}\sum_{p=1}^{h'w'} \mathbf{W}_p^{(q)}$. Since we assume that the weights of the linear layer are i.i.d. $\mathcal{N}\left(0, \tau^2\right)$, for the upper part, and 0 for the lower part, α_q is a centered Gaussian with variance $(\frac{\tau}{hw})^2 \frac{h'w'}{2}$. Now, we recall that $\mathbf{B}$ does not depend on $\mathbf{W}$. Therefore, conditionally to $\mathbf{F}^{(q)}$, $\alpha_q \mathbf{B}_{i,j}^{(q)}$ is a centered Gaussian with variance

$$\left(\mathbf{B}_{i,j}^{(q)}\right)^2 \left(\frac{\tau}{hw}\right)^2 \frac{h'w'}{2}.$$

We deduce that

$$\sum_{q=1}^{V} \alpha_q \mathbf{B}_{i,j}^{(q)} | \mathbf{F} \sim \mathcal{N}\left(0, \sum_{q=1}^{V} \left(\mathbf{B}_{i,j}^{(q)}\right)^2 \left(\frac{\tau}{hw}\right)^2 \frac{h'w'}{2}\right). \tag{11}$$

Computing expectations. Since, conditionally to $\mathbf{F}$, $\sum_q \alpha_q \mathbf{B}^{(q)}$ is a a centered Gaussian with variance given by Eq. (11), we can use Lemma 1 to compute

$$\begin{aligned}\mathbb{E}\left[\sigma\left(\sum_{q=1}^{V} \alpha_q \mathbf{B}_{i,j}^{(q)}\right)\middle|\mathbf{F}\right] &= \sqrt{\left(\frac{\tau}{hw}\right)^2 \frac{h'w'}{2} \sum_{q=1}^{V} \left(\mathbf{B}_{i,j}^{(q)}\right)^2}\,\phi(0) \\ &= \frac{\tau}{2hw\sqrt{\pi}} \sqrt{h'w' \sum_{q=1}^{V} \left(\mathbf{B}_{i,j}^{(q)}\right)^2}.\end{aligned}$$

Coming back to Eq. (10), we deduce that

$$\mathbb{E}\left[\sigma\left(\sum_{q=1}^{V} \alpha_q \mathbf{B}_{i,j}^{(q)}\right)\right] = \frac{\tau}{2hw\sqrt{\pi}} \sqrt{h'w'}\, \mathbb{E}\left[\sqrt{\sum_{q=1}^{V} \left(\mathbf{B}_{i,j}^{(q)}\right)^2}\right]. \tag{12}$$

Lower bound. Up to the best of our knowledge, there is no closed-form expression for Eq. (12), and we now proceed to find a lower bound to this expression. Let $Y := \sum_{q=1}^{V} \left(\mathbf{B}_{i,j}^{(q)}\right)^2$. By Lemma 2 and 3, $\mathbb{E}[Y] = V\frac{(\tau\|\mathbf{m}\|_2)^2}{2}$ and $\mathrm{Var}(Y) = \frac{5}{4}V(\tau\|\mathbf{m}\|_2)^4$ with $\mathbf{m} := \boldsymbol{\xi}_{i:i+k-1,j:j+k-1}$ a patch at index (i,j) in the image $\boldsymbol{\xi}$. By Chebyshev's inequality, for any $t > 0$,

$$\mathbb{P}(Y \geqslant \mathbb{E}[Y] - t) \geqslant 1 - \frac{\mathrm{Var}(Y)}{t^2}. \tag{13}$$

Let us set $t = \frac{\mathbb{E}[Y]}{2} > 0$ in the previous display, in this way

$$\frac{\mathrm{Var}(Y)}{t^2} = \frac{5}{4}V\tau^4\|m\|^4 \cdot \left(\frac{V^2\tau^4\|m\|^4}{2^2\cdot 2^2}\right)^{-1} = \frac{20}{V}.$$

Since $\|\mathbf{m}\|_2 > 0$,

$$\mathbb{P}\left(Y \geqslant \frac{\mathbb{E}[Y]}{2}\right) \geqslant 1 - \frac{20}{V}.$$

We now conclude the proof writing

$$\begin{aligned}
&\mathbb{E}\left[\sigma\left(\sum_{q=1}^{V}\alpha_q\mathbf{B}_{i,j}^{(q)}\right)\right] \\
&= \frac{\tau}{2hw\sqrt{\pi}}\sqrt{h'w'}\left(\mathbb{E}\left[\sqrt{Y}\middle| Y \geqslant \frac{\mathbb{E}[Y]}{2}\right]\mathbb{P}\left(Y \geqslant \frac{\mathbb{E}[Y]}{2}\right)\right. \\
&\quad \left. + \mathbb{E}\left[\sqrt{Y}\middle| Y < \frac{\mathbb{E}[Y]}{2}\right]\mathbb{P}\left(Y < \frac{\mathbb{E}[Y]}{2}\right)\right) \\
&\geqslant \frac{\tau}{2hw\sqrt{\pi}}\sqrt{h'w'}\mathbb{E}\left[\sqrt{Y}\middle| Y \geqslant \frac{\mathbb{E}[Y]}{2}\right]\mathbb{P}\left(Y \geqslant \frac{\mathbb{E}[Y]}{2}\right) \qquad \text{(since } Y > 0\text{)} \\
&\geqslant \frac{\tau}{2hw\sqrt{\pi}}\sqrt{h'w'}\,\mathbb{E}\left[\sqrt{\frac{\mathbb{E}[Y]}{2}}\right]\left(1 - \frac{20}{V}\right) \\
&= \frac{\sqrt{Vh'w'}}{4\sqrt{\pi}}\frac{\tau^2}{hw}\left(1 - \frac{20}{V}\right)\|m\|_2.
\end{aligned}$$

□

E Additional Experiments

In this section, we present an additional set of experiments on STACK-MIX (Figure 9).

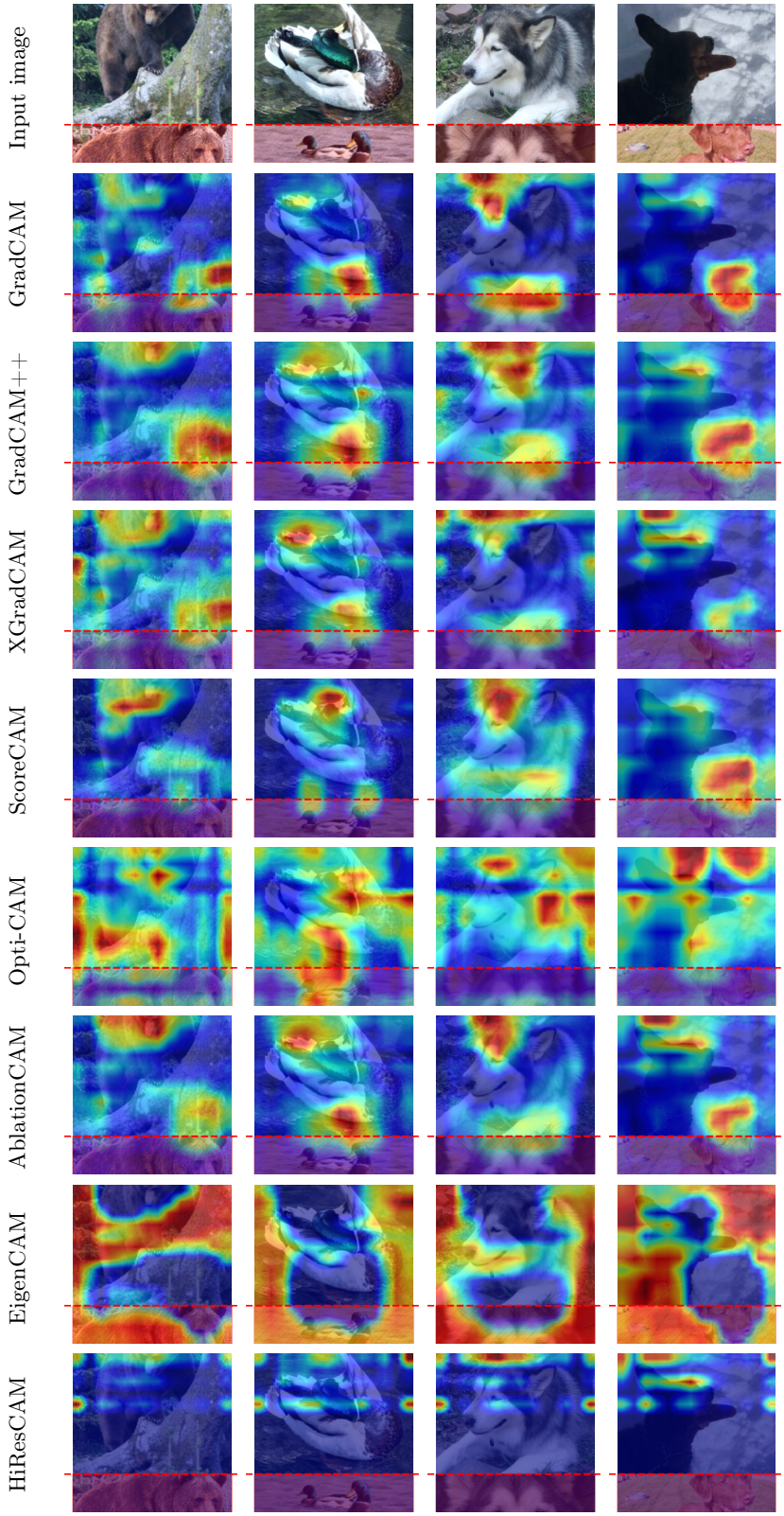

Fig. 9: Saliency maps given by the considered CAM-based methods for [**VGG**]. With the notable exception of HiresCAM, they all highlight parts of images from STACK-MIX which are unseen by the network (this is denoted by the red, rectangular shape in the lower part of the image).