

There and Back Again: Revisiting Backpropagation Saliency Methods

Sylvestre-Alvise Rebuffi* Ruth Fong* Xu Ji* Andrea Vedaldi
 Visual Geometry Group, University of Oxford
 {srebuffi, ruthfong, xuji, vedaldi}@robots.ox.ac.uk

Abstract

Saliency methods seek to explain the predictions of a model by producing an importance map across each input sample. A popular class of such methods is based on backpropagating a signal and analyzing the resulting gradient. Despite much research on such methods, relatively little work has been done to clarify the differences between such methods as well as the desiderata of these techniques. Thus, there is a need for rigorously understanding the relationships between different methods as well as their failure modes. In this work, we conduct a thorough analysis of backpropagation-based saliency methods and propose a single framework under which several such methods can be unified. As a result of our study, we make three additional contributions. First, we use our framework to propose NormGrad, a novel saliency method based on the spatial contribution of gradients of convolutional weights. Second, we combine saliency maps at different layers to test the ability of saliency methods to extract complementary information at different network levels (e.g. trading off spatial resolution and distinctiveness) and we explain why some methods fail at specific layers (e.g., Grad-CAM anywhere besides the last convolutional layer). Third, we introduce a class-sensitivity metric and a meta-learning inspired paradigm applicable to any saliency method for improving sensitivity to the output class being explained.

1. Introduction

The adoption of deep learning methods by high-risk applications, such as healthcare and automated driving, gives rise to a need for tools that help machine learning practitioners understand model behavior. Given the highly-parameterized, opaque nature of deep neural networks, developing such methods is non-trivial, and there are many possible approaches. In the basic case, even the predictions of the model itself, either unaltered or after being distilled into a simpler function [11, 3], can be used to shed light on

its behaviour.

Saliency is the specific branch of interpretability concerned with determining not what the behaviour of a model is for whole input samples, but which parts of samples contribute the most to that behaviour. Thus by definition, determining saliency - or attribution - necessarily involves reversing the model's inference process in some manner [20]. Propagating a signal from the output layer of a neural network model back to the input layer is one way of explicitly achieving this.

The number of diverse works based on using signal backpropagation for interpretability in computer vision [37, 27, 4, 38] is testimony to the power of this simple principle. Typically, these techniques produce a heatmap for any given input image that ranks its pixels according to some metric of importance for the model's decision. Inspired by the work of [1], we propose to delve deeper into such methods by discussing some of the similarities, differences and potential improvements that can be illustrated.

We begin with presenting a framework that unifies several backpropagation-based saliency methods by dividing the process of generating a saliency map into two phases: *extraction* of the contribution of the gradient of network parameters at each spatial location in a particular network layer, and *aggregation* of such spatial information into a 2D heatmap. GradCAM [27], linear approximation [17] and gradient [29] can all be cast as such processes. Noting that no appropriate technique has yet been proposed for properly aggregating contributions from convolutional layers, we introduce NormGrad, which uses the Frobenius norm for aggregation. We introduce identity layers to allow for the generation of saliency heatmaps at all spatially-grounded layers in the network (i.e. even after ReLU), since NormGrad computes saliency given a parameterised network layer.

We conduct a thorough analysis of backpropagation-based saliency methods in general, with evaluation based on utilising saliency heatmaps for weak localisation. Notably, we conduct an investigation into simple techniques for combining saliency maps taken from different network layers - in contrast to the popular practice of computing maps at the input layer [29] - and find that using a weighted aver-

*indicates equal contribution

age of maps from all layers consistently improves performance for several saliency methods, compared to taking the single best layer. However, not all layers are equally important, as we discover that models optimised on datasets such as ImageNet [26] and PASCAL VOC [7] learn features that become increasingly *class insensitive* closer to towards the input layer. This provides an explanation for why Grad-CAM [27] produces unmeaningful saliency heatmaps at certain layers earlier than the last convolutional layer, as the sensitivity of the gradient to class across spatial locations is eliminated by Grad-CAM’s spatial gradient averaging, meaning such layers are devoid of class sensitive signals from which to form saliency heatmaps.

Finally, building off [20, 1], we introduce a novel metric for quantifying the class sensitivity of a saliency method, and present a meta-learning inspired paradigm that increases the class sensitivity of any method by adding an inner SGD step into the computation of the saliency heatmap.

2. Related work

Saliency methods. Our work focuses on backpropagation-based saliency methods; these techniques are computationally efficient as they only require one forward and backward pass through a network. One of the earliest methods was [29] which visualised the gradient at the input with respect to an output class being explained. Several authors have since proposed adaptations in order to improve the heatmap’s visual quality. These include modifying the ReLU derivatives (Deconvnet [37], guided backprop [32]) and averaging over randomly perturbed inputs (SmoothGrad [31]) to produce masks with reduced noise. Several works have explored visualizing saliency at intermediate layers by combining information from activations and gradients, notably CAM [39], Grad-CAM [27], and linear approximation (a.k.a. gradient \times input) [17]. Conservation-preserving methods (Excitation Backprop [38], LRP [4], and DeepLIFT [28]) modify the backward functions of network layers in order to “preserve” an attribution signal such that it sums to one at any point in the network. Reference-based methods average over attributions from multiple interpolations [33] between the input and a non-informative (e.g. black) reference input or use a single reference input with which to compare a backpropagated attribution signal [28].

Although we focus on backpropagation-based methods, another class of methods studies the effects that perturbations on the input induce on the output. [37] and [24] generate saliency maps by weighting input occlusion patterns by the induced changes in model output. [10, 9, 15] learn for saliency maps that maximally impact the model, and [5] trains a model to predict effective maps. LIME [25] learns linear weights that correspond to the effect of including or excluding (via perturbation) different image patches in an

image. Perturbation-based approaches have also been used to perform object localisation [30, 35, 34].

Assessing and unifying saliency methods. A few works have studied if saliency methods have certain desired sensitivities (e.g., to specific model weights [1] or the output class being explained [20]) and if they are invariant to unmeaningful factors (e.g., constant shift in input intensity [16]). [20] showed that gradient [29], deconvnet [37], and guided backprop [32] tend to produce class insensitive heatmaps. [1] introduced sanity check metrics that measure how sensitive a saliency method is to model weights by reporting the correlation between a saliency map on a trained model vs. a partially randomized model.

Other works quantify the utility of saliency maps for weak localization [38, 10] and for impacting model predictions. [38] introduced Pointing Game, which measures the correlation between the maximal point extracted from a saliency map with pixel-level semantic labels. [10] extracts bounding boxes from saliency maps and measures their agreement with ground truth bounding boxes. [36] evaluates attribution methods using relative feature importance. [22] proposes a dataset designed for measuring visual explanation accuracy. [4, 24, 15] present variants of a perturbation-based evaluation metric that measures the impact of perturbing (or unperturbing for [15]) image patches in order of importance as given by a saliency map. However, these perturbed images are outside the training domain; [13] mitigates this by measuring the performance of classifiers re-trained on perturbed images (i.e., with 20% of pixels perturbed).

To our knowledge, the only work that has been done to unify saliency methods focuses primarily on “invasive” techniques that change backpropagation rules. The α -LRP variant [4] and Excitation Backprop [38] share the backpropagation rule, and DeepLIFT [28] is equivalent to LRP when 0 is used as the reference activation throughout a network. [19] unifies a few methods (e.g., LIME [25], LRP [4], DeepLIFT [28]) under the framework of additive feature attribution.

3. Method

Preliminaries. Consider a training set \mathcal{D} of pairs (\mathbf{x}, \mathbf{y}) where $\mathbf{x} \in \mathbb{R}^{3 \times H \times W}$ are (color) images and $\mathbf{y} \in \{1, \dots, C\}$ their labels. Furthermore, let $\mathbf{y} = \Phi_{\theta}(\mathbf{x})$ be the output of a neural network model whose parameters θ are optimized using the cross-entropy loss ℓ to predict labels from images.

3.1. Extract & Aggregate framework

In most methods, the saliency map is obtained as a function of the network activations, computed in a forward pass,

and information propagated from the output of the network back to its input using the backpropagation algorithm. While some methods modify backpropagation in some way, here we are interested in those, such as gradient, linear approximation, and all variants of our proposed NormGrad saliency method, that do not.

In order to explain these “non-invasive” methods, we suggest that their saliency maps can be interpreted as a measure of how much the corresponding pixels contribute to changing the model parameters during a training step. We then propose a principled two-phase framework capturing this idea. In the extraction phase, a method isolates the contribution to the gradient from each spatial location. We use the fact that the gradient of spatially shared weights can be written as the sum over spatial locations of a function of the activation gradients and input features. In the aggregation phase, each spatial summand is converted into a scalar using an aggregation function, thus resulting in a saliency map.

Algorithm 1 Extract & Aggregate

1. **Extract.** Compute spatial contributions to the summation of the gradient of the weights.
 - Choose a layer whose parameters are shared spatially (layers from table 1).
 - Alternatively, insert an identity layer (section 3.1.2) at the targeted location.
 2. **Aggregate.** Transform these spatial contributions into saliency maps using an aggregation function.
 - Norm: NormGrad (ours)
 - Voting/Summing: linear approximation [17]
 - Filtering: GradCAM [27], selective NormGrad (ours)
 - Max: Gradient backpropagation [29]
-

3.1.1 Phase 1: Extract

We first choose a target layer in the network at which we plan to compute a saliency map. Assuming that the network is a simple chain¹, we can write $L = \ell \circ \Phi = h \circ k_w \circ q$, where k_w is the target layer parameterised by w , h is the composition of all layers that follow it (including loss ℓ), and q is the composition of layers that precede it. Then, $\mathbf{x}^{in} = q(\mathbf{x}) \in \mathbb{R}^{K \times H \times W}$ denotes the input to the target layer, and its output is given by $\mathbf{x}^{out} = k_w(\mathbf{x}^{in}) \in \mathbb{R}^{K' \times H' \times W'}$.

¹Other topologies are treated in the same manner, but the notation is more complex.

| Layer | Spatial contribution | Size at each location |
|-------------------|--|---------------------------|
| Bias | \mathbf{g}_u^{out} | vector: K' |
| Scaling | $\mathbf{g}_u^{out} \odot \mathbf{x}_u^{in}$ | vector: K' |
| Conv $N \times N$ | $\mathbf{g}_u^{out} \mathbf{x}_{u,N \times N}^{in \top}$ | matrix: $K' \times N^2 K$ |

Table 1. Formulae and sizes of the spatial contributions to the gradient of the weights for layers with spatially shared parameters. \odot denotes the elementwise product and $\mathbf{x}_{u,N \times N}^{in}$ is the $N^2 K$ vector obtained by unfolding the $N \times N$ patch around the target location.

Convolutional layers with general filter shapes. For convolutions with an $N \times N$ kernel size, we can re-write the convolution using the matrix form:

$$\mathbf{X}^{out} = \mathbf{W} \mathbf{X}_{N \times N}^{in} \quad (1)$$

where $\mathbf{X}^{out} \in \mathbb{R}^{K' \times HW}$ and $\mathbf{W} \in \mathbb{R}^{K' \times N^2 K}$ are the output and filter tensors reshaped as matrices and $\mathbf{X}_{N \times N}^{in} \in \mathbb{R}^{N^2 K \times HW}$ is a matrix whose column $\mathbf{x}_{u,N \times N}^{in} \in \mathbb{R}^{N^2 K}$ contains the unfolded patches at location u of the input tensor to which filters are applied.² Then, the gradient w.r.t. the filter weights W is given by

$$\frac{dL}{d\mathbf{W}} = \sum_{u \in \Omega} \frac{d}{d\mathbf{W}} \langle \mathbf{g}_u^{out}, \mathbf{W} \mathbf{x}_{u,N \times N}^{in} \rangle = \sum_{u \in \Omega} \mathbf{g}_u^{out} \mathbf{x}_{u,N \times N}^{in \top}, \quad (2)$$

where $\mathbf{g}_u^{out} = dh/d\mathbf{x}_u^{out}$ is the gradient of the “head” of the network. Thus, for the convolutional layer case, the spatial summand is an outer product of two vectors; thus, the spatial contribution at each location to the gradient of the weights is a matrix of size $K' \times N^2 K$.

Other layer types. Besides convolutional layers, bias and scaling layers also share their parameters spatially. In modern architectures, these are typically used in batch normalization layers [14]. We denote $\mathbf{b}, \alpha \in \mathbb{R}^K$ as the parameters for the bias and scaling layers respectively. They are defined respectively as follows:

$$x_{ku}^{out} = x_{ku}^{in} + b_k, \quad x_{ku}^{out} = \alpha_k x_{ku}^{in}.$$

Then, the gradients w.r.t. parameters are given by

$$\frac{dL}{d\mathbf{b}} = \sum_{u \in \Omega} \mathbf{g}_u^{out}, \quad \frac{dL}{d\alpha} = \sum_{u \in \Omega} \mathbf{g}_u^{out} \odot \mathbf{x}_u^{in}. \quad (3)$$

where \odot is the elementwise product. For these two types of layer, the spatial summand is a vector of size K , the number of channels. Table 1 summarizes the spatial contributions to the gradients for the different layers.

²This operation is called `im2row` in MATLAB or `unfold` in PyTorch.

3.1.2 Virtual identity layer

So far, we have only extracted spatial gradient contributions for layers that share parameters spatially. We will now extend our summand extraction technique to any location within a CNN by allowing the insertion of a virtual identity layer at the target location. This layer is a conceptual construction that we introduce to derive in a rigorous and unified manner the equations employed by various methods to compute saliency.

We are motivated by the following question: if we were to add an identity operator at a target location, how should this operator's parameters be changed? A virtual identity layer is a layer which shares its parameters spatially and is set to the identity. Hence, it could be any of the layer from table 1, like a bias or a scaling layer; then, $b_k = 0$ or $c_k = 1$ respectively for $k \in \{1, \dots, K\}$. It could also be a $N \times N$ convolutional layer with filter bank $\mathbf{W} \in \mathbb{R}^{K \times N^2 K}$ such that $w_{kk'ij} = \delta_{k,k'} \delta_{i,0} \delta_{j,0}$, where $\delta_{a,b}$ is the Kronecker delta function³, for $(i, j) \in \{-\frac{N-1}{2}, \dots, \frac{N-1}{2}\}$.

Because of the nature of the identity, this layer does not change the information propagated in either the forward or backward direction. Conceptually, it is attached to the part of the model that one wishes to inspect as shown in fig. 1. This layer is never “physically” added to the model (i.e., the model is not modified); its “inclusion” or “exclusion” simply denotes whether we are using input or output activations (\mathbf{x}^{in} or \mathbf{x}^{out}). Indeed the backpropagated gradient \mathbf{g}^{out} at the output of the identity as shown in fig. 1 is the gradient that would have been at the output of the layer preceding the identity. This construction allows to examine activations and gradients at the same location (e.g., \mathbf{x}^{out} and \mathbf{g}^{out}) as most existing saliency methods do. We now can use the formulae defined in section 3.1.1 when analyzing the gradient of the weights of this identity layer. For example, if we consider an identity scaling layer, the spatial contribution would be $\mathbf{g}_u^{out} \odot \mathbf{x}_u^{out}$. Or, for a $N \times N$ identity convolutional layer, it would be $\mathbf{g}_u^{out} \mathbf{x}_{u,N \times N}^{out \top}$.

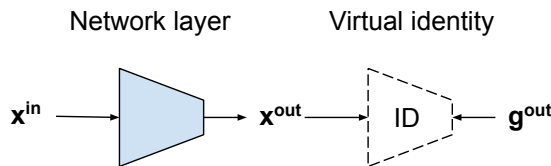


Figure 1. **Virtual identity.** Here, we visualize inserting an identity after a specific layer in the network for saliency computation purposes. The gradient coming from later stages of the network that is the gradient at output of the network layer, is now also the gradient at the output of the identity, \mathbf{g}^{out} .

³Kronecker delta function: $\delta_{a,b} = 1$ if $a = b$; otherwise, $\delta_{a,b} = 0$.

3.1.3 Phase 2: Aggregate

Following the extraction phase (section 3.1.1), we have the local contribution at each spatial location to the gradients of either an existing layer or a virtual identity layer. Each spatial location is associated with a vector or matrix (table 1). In this section, we describe different aggregation functions to map these vectors or matrices to a single scalar per spatial position. We drop the spatial location u in the notations.

Understanding existing saliency methods. One possible aggregation function is the sum of the elements in a vector. When combined with a virtual scaling identity layer (section 3.1.2), we obtain the linear approximation method [17]: $\sum_k g_k^{out} x_k^{out}$. The contributions from the scaling identity encode the result of channels changing (after the gradient is applied) at each location; thus, the sum aggregation function acts as a voting mechanism. The resulting saliency map highlights the locations that would be most impacted if following through with the channel updates.

Aggregation functions can also be combined with element-wise filtering functions (e.g., the absolute value function). Another aggregation function takes the maximum absolute value of the vector: $\max_{abs}(\mathbf{x}) = \max_k |x_k|$. If we combine a virtual bias identity layer in phase 1, which gives \mathbf{g}^{out} as the spatial contribution, with the maxabs function for aggregation, we obtain at each spatial location the gradient [29] method: $\max_k |g_k^{out}|$.

As for CAM [39] and Grad-CAM [27], we cannot directly use the spatial contributions extracted at each location because they spatially average \mathbf{g}^{out} .⁴ However, for architectures that use global average pooling followed after their convolutional layers (e.g., ResNet architectures), $\mathbf{g}_u^{out} = \bar{\mathbf{g}}^{out}$. Then, CAM and Grad-CAM and CAM can be viewed as combining a virtual scaling identity layer from phase 1 with summing and positive filtering (i.e., $\text{filter}_+(x) = (x)_+$) functions for aggregation.

NormGrad. The sum and max functions have clear interpretations when using bias or scaling identity layers; however, they cannot be easily transported to convolutional identity layers as interactions between input channels can vary depending on the output channel and are represented as a matrix, not as a vector as are the case for scaling and bias layers. Thus, we would like to have an aggregation function that could be used to aggregate any type of spatial contribution, regardless of its shape.

Using the norm function satisfies this criterion, for example the L^2 norm when dealing with vectors and the Frobenius norm for matrices. We note that the matrices obtained at each location for convolutional layers are the outer

⁴Because CAM global average pooling + one fully connected layer, \mathbf{g}^{out} is equal to the fully connected weights.

| Phase 1 | Phase 2 | Saliency map |
|------------------------|---------|---------------------------------------|
| Bias IDL | Max | $\max_k g_k^{out}$ |
| Scaling IDL | Sum | $\sum_k g_k^{out} x_k^{out}$ |
| Scaling IDL | F + N | $\ (g^{out} \odot x^{out})_+\ $ |
| Conv 1×1 IDL | Norm | $\ g^{out}\ \ x^{out}\ $ |
| Real Conv 3×3 | Norm | $\ g^{out}\ \ x_{3 \times 3}^{in}\ $ |

Table 2. **Combining layers and aggregation functions for saliency.** g_k^{out} and x_k^{out} are tensor slices for channel k and contain only spatial information. IDL denotes an identity layer. F + N is positive filtering + norm. From top to bottom, the rows correspond to the following saliency methods: 1., gradient, 2., linear approximation, 3., NormGrad selective, 4., NormGrad, and 5., NormGrad without the virtual identity trick.

products of two vectors. For such matrices, the Frobenius norm is equal to the product of the norms of the two vectors. For example, for an existing 1×1 convolutional layer, we consider the saliency map given by $\|g^{out}\| \|x^{in}\|$. We call this class of saliency methods derived by using norm as an aggregation function, “NormGrad”.

Saliency maps from the NormGrad outlined above are not as class selective as other methods because they highlight the spatial locations that contribute the most to gradient of the weights, both positively and negatively. One way to introduce class selectivity is to use positive filtering before applying the norm. If we apply norm and positive filtering aggregation to a scaling identity layer, the resulting saliency map is given by $\|(g^{out} \odot x^{out})_+\|$. Throughout the rest of the paper, we call this variant “selective NormGrad”.

4. Experiments

In this section we quantitatively and qualitatively evaluate the performance of a large number of backpropagation-based saliency methods. Code for our framework is released at http://github.com/srebuffi/revisiting_saliency. Additional experiments such as image captioning visualizations are included in the appendix.

Experimental set-up. Unless otherwise stated, saliency maps are generated on images from either the PASCAL VOC [7] 2007 test set or the ImageNet [6] 2012 validation set for either VGG16 [29] or Resnet50 [12]. For PASCAL VOC, we use a model pre-trained on ImageNet with fine-tuned fully connected layers on PASCAL VOC. We use [9]’s TorchRay interpretability package to generate saliency methods for all other saliency methods besides our NormGrad and meta-saliency methods as well as to evaluate saliency maps on [38]’s Pointing Game (see [38] for more details). For all correlation analysis, we compute the Spearman’s correlation coefficient [21] between saliency maps that are upsampled to the input resolution: 224×224 .

4.1. Justifying the virtual identity trick.

In order to justify our use of the virtual identity trick, we compare NormGrad saliency maps generated at 1×1 and 3×3 convolutional layers both with and without the virtual identity trick (4th and 5th rows respectively in table 2) for VGG16 and ResNet50. We first computed the correlations between saliency maps generating with and without the virtual identity trick. We found that the mean correlation across $N = 50k$ ImageNet validation images was over 95% across all layers we tested. We also evaluated their performance on the Pointing Game [38] and found that the mean absolute difference in pointing game accuracy was $0.53\% \pm 0.62\%$ across all layers we tested (see supp. for more details and full results). This empirically demonstrates that using the virtual identity trick closely approximates the behavior of calculating the spatial contributions for the original convolutional layers.

4.2. Combining saliency maps across layers.

Training linear classifiers on top of intermediate representations is a well-known method for evaluating the learned features of a network at different layers [2]. This suggests that saliency maps, too, may have varying levels of meaningfulness at different layers.

We explore this question by imposing several weighting methods for combining the layer-wise saliency maps of ResNet50 and VGG16 and measuring the resulting performance on PASCAL VOC Pointing Game on both the “difficult” and “all” image sets. To determine the weight γ_j for a given layer j out of J layers in a network we use:

1. **Feature spread.** Given the set of feature activations at layer j , x^i for $i \in M$ input images sampled uniformly across classes, compute the spatial mean $\bar{x}^i = \sum_{u \in \Omega} x_u^i$. $\gamma_j = \frac{1}{M} \sum_{i=1}^M |\bar{x}^i - \bar{x}^\mu|$ where $\bar{x}^\mu = \frac{1}{M} \sum_{i=1}^M |\bar{x}^i|$.
2. **Classification accuracy.** Given the set of feature activations at layer j , x^i for $i \in M$ input images sampled uniformly across classes, train a linear layer Ψ using image labels y^i . $\gamma_j = \frac{1}{M} \sum_{i=1}^M \delta_{\Psi(x^i), y^i}$ where $\delta_{\Psi(x^i), y^i} = 1$ if $\Psi(x^i) = y^i$, 0 otherwise.
3. **Linear interpolation.** $\gamma_j = \frac{j}{J}$.
4. **Uniform.** $\gamma_j = \frac{1}{J}$.

To obtain a combined saliency map M from maps m_j from each layer j , the weights are normalised and applied either additively, $M = \sum_{j=0}^J \gamma_j \cdot m_j$, or with a product, $M = \prod_{j=0}^J m_j^{\gamma_j}$ (see fig. 2 for visualization).

Table 3 shows that weighted saliency maps produce the best overall performance in all four key test cases, which is surprising as there were only four weight schemes tested

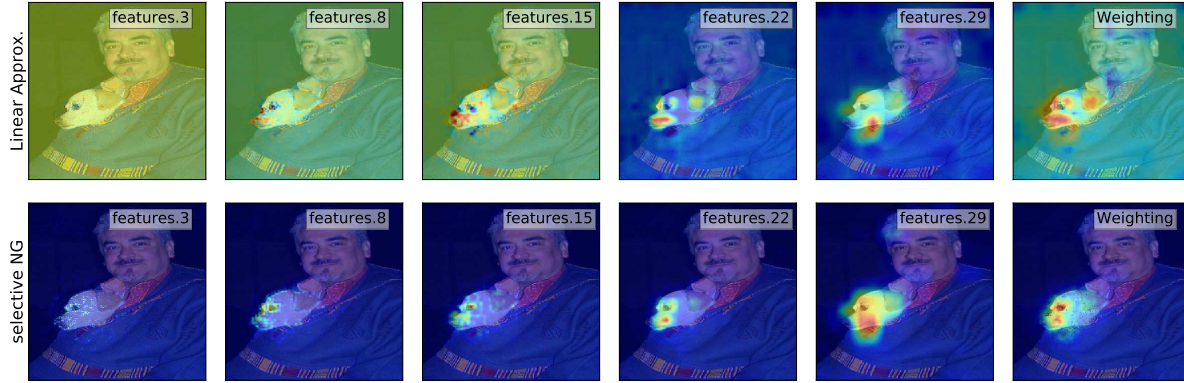


Figure 2. **Saliency maps at different network depths and as a weighted combination.** Linear approximation vs. selective NormGrad saliency maps of VGG16 on VOC2007. The first 5 images of each row correspond to different depths whereas the last one is a weighted product combination (using classification accuracy weights) of the first saliency maps. We observe that the weighted version produces more fine grained maps for both methods.

| | All | | | | Difficult | | | |
|-----|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | Resnet50 | | VGG16 | | Resnet50 | | VGG16 | |
| | b.s. | b.w. | b.s. | b.w. | b.s. | b.w. | b.s. | b.w. |
| CEB | 90.7 | 88.6 | 82.1 | 78.2 | 82.2 | 82.2 | 67.0 | 65.2 |
| EB | 84.5 | 83.1 | 77.5 | 75.7 | 71.5 | 71.3 | 57.8 | 56.1 |
| GC | 90.3 | 90.5 | 86.6 | 80.6 | 82.3 | 82.6 | 74.0 | 67.8 |
| Gd | 83.9 | 83.3 | 86.6 | 82.7 | 70.3 | 69.4 | 66.4 | 67.4 |
| Gds | 80.0 | 77.4 | 76.8 | 77.2 | 62.9 | 59.5 | 57.9 | 59.4 |
| Gui | 82.3 | 81.0 | 75.8 | 74.4 | 67.9 | 63.4 | 53.0 | 51.6 |
| LA | 90.2 | 91.2 | 86.4 | 86.9 | 81.9 | 83.8 | 74.5 | 77.4 |
| NG | 84.6 | 83.5 | 81.9 | 81.8 | 72.2 | 70.2 | 64.8 | 64.6 |
| sNG | 87.4 | 88.7 | 86.0 | 86.8 | 77.0 | 79.1 | 72.6 | 74.5 |

Table 3. **Pointing game results on VOC07.** b.s. and b.w. stand for best single layer and best weighted combination. (C)EB: (Contrastive) Excitation Backprop, GC: GradCAM, Gd(s): Gradient (sum), Gui: guided backprop, LA: linear approximation, (s)NG: (selective) NormGrad.

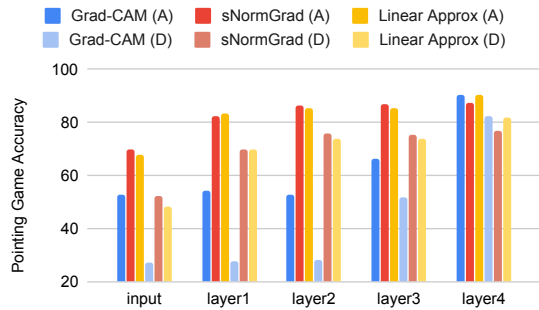


Figure 3. **Select Pointing Game results.** Results for ResNet50 on VOC07 at different network depths (A: all images; D: difficult subset). Grad-CAM performs worse at every layer except the last conv layer and lower than pointing at the center (all: 69.6%; diff: 42.4%) at most layers.

(in addition to best single layer), none of which were explicitly optimised for use with saliency maps. Our results strongly indicate that linear approximation in particular benefits from combining maps from different layers, and linear approximation with layer combination consistently produces the best performance overall and beats far more complex methods at weak localisation using a single forward-backward pass (see supp. for full results).

Note that the feature spread and classification accuracy metrics can both be used as indicators of class sensitivity (section 4.3). This is because if feature activations are uniform for images sampled across classes, it is not possible for them to be sensitive to - or predictive of - class, and the classification accuracy metric is an explicit quantisation of how easily features can be separated into classes. We observe from the computed weights that both metrics generally increase with layer depth (see supp.).

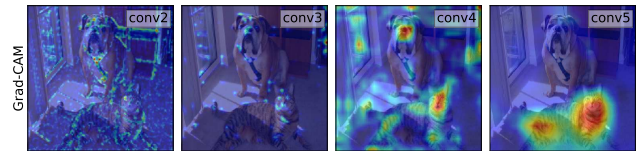


Figure 4. **Grad-CAM failure mode.** Grad-CAM saliency maps w.r.t. “tiger cat” at different depths of VGG16. Grad-CAM only works at the last conv layer (rightmost col).

Explanation of Grad-CAM failure mode. Figure 4 showed qualitatively that Grad-CAM does not produce meaningful saliency maps at any layer except the last convolutional layer, which is confirmed by Grad-CAM’s Pointing Game results at earlier layers. Class sensitivity - as measured by our weighting metrics - increasing with layer depth offers an explanation for this drop in performance. Since

Grad-CAM spatially averages the backpropagated gradient before taking a product with activations, each pixel location in the heatmap receives the same gradient vector (across channels) *irrespective* of the image content contained within its receptive field. Thus, if the activation map used in the ensuing product is also not class selective - firing on both dogs and cats for example, fig. 4 - the saliency map cannot be. On the other hand, methods that do not spatially average gradients such as NormGrad (fig. 3) can rely on gradients that are free to vary across the heatmap with underlying class, increasing the class sensitivity of the resulting saliency map.

4.3. An explicit metric for class sensitivity

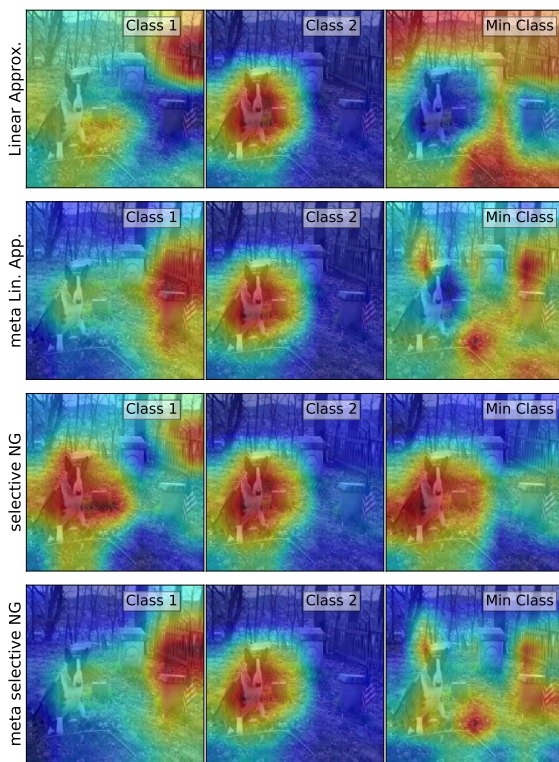


Figure 5. **Class sensitivity with and without meta-saliency.** Min class saliency maps that use meta-saliency (row 2 and 4, right col) are less informative than those that don't use meta-saliency (rows 1 and 3, right col). Class 1 is the ground truth class (fence), class 2 is the maximally predicted class (Cardigan Welsh corgi), min class is the minimally predicted class (black widow spider).

[20] qualitatively shows that early backprop-based methods (e.g., gradient, deconvnet, and guided backprop) are not sensitive to the output class being explained by showing that saliency maps generated w.r.t. different output classes and gradient signals appear visually indistinguishable. Thus, similar to [1], we introduce a sanity check to measure a saliency method's output class sensitivity. We compute the correlation between saliency maps w.r.t. to output class pre-

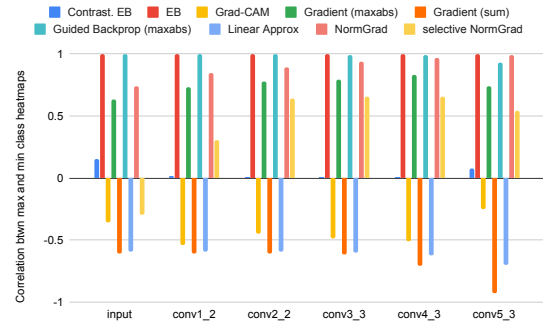


Figure 6. **Class sensitivity of saliency methods.** This plot shows the correlation between VGG16 saliency maps computed w.r.t. to the maximally and minimally predicted class (closer to zero is better).

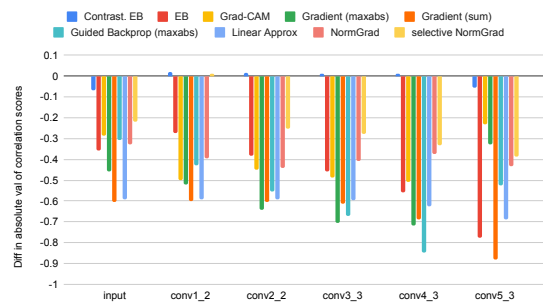


Figure 7. **Meta-saliency improves class sensitivity for all saliency methods.** Using meta-saliency yields weaker correlations between the saliency maps w.r.t. the maximally and minimally predicted output class compared to not using meta-saliency (lower is better).

dicted with highest confidence (max class) and that predicted with lowest confidence (max and min class respectively) for $N = 1000$ ImageNet val. images (1 per class).

We would expect saliency maps w.r.t. the max class to be visually salient while those w.r.t. to the min class to be uninformative (because the min class is not in the image). Thus, we desire the correlation scores to be close to zero.

Figure 6 shows results for various saliency methods. We observe that excitation backprop and guided backprop yield correlation scores close to 1 for all layers, while contrastive excitation backprop yields scores closest to 0. Furthermore, methods using sum aggregation (e.g., gradient [sum], linear approx, and Grad-CAM) have negative scores (i.e., their max-min-class saliency maps are anti-correlated). This is because sum aggregation acts as a voting mechanism; thus, these methods reflect the fact that the network has learned anti-correlated relationships between max and min classes.

4.4. Meta-saliency analysis

As a general method for improving the sensitivity of saliency heatmaps to the output class used to generate the

gradient, we propose to perform an inner SGD step before computing the gradients with respect to the loss. This way we can extend any saliency method to second order gradients. This is partly inspired by the inner step used in, for example, few shot learning [8] and architecture search [18]. We want to minimize:

$$L(\theta, x) = \ell(\theta - \epsilon \nabla_{\theta} \ell(\theta, x), x). \quad (4)$$

We take $\epsilon \ll 1$ to use a Taylor expansion of this loss at θ and we now have the resulting approximated loss:

$$L(\theta, x) \approx \ell(\theta, x) - \epsilon \|\nabla_{\theta} \ell(\theta, x)\|^2. \quad (5)$$

As done in the previous section, we can now take the gradient of the loss with respect to the parameters θ :

$$\nabla_{\theta} L(\theta, x) \approx \nabla_{\theta} \ell(\theta, x) - 2\epsilon \nabla_{\theta}^2 \ell(\theta, x) \nabla_{\theta} \ell(\theta, x). \quad (6)$$

Using a finite difference scheme of step h as in [23], we can approximate the hessian-vector product by:

$$\nabla_{\theta}^2 \ell(\theta, x) \nabla_{\theta} \ell(\theta, x) = \frac{\nabla_{\theta} \ell(\theta, x) - \nabla_{\theta} \ell(\theta^-, x)}{h} + O(h).$$

where $\theta^- = \theta - h \nabla_{\theta} \ell(\theta, x)$. We chose on purpose a backward finite difference such that two terms cancel each other when taking $h = 2\epsilon$ and we get:

$$\nabla_{\theta} L(\theta, x) \approx \nabla_{\theta'} \ell(\theta', x).$$

where $\theta' = \theta - 2\epsilon \nabla_{\theta} \ell(\theta, x)$ corresponds to one step of SGD of learning rate 2ϵ . We notice that if we take $\epsilon \rightarrow 0$, this formula boils back down to the original gradient of the weights without meta step. We further note that this meta saliency approach only requires one more forward-backward pass compared to usual saliency backpropagation methods.

Conversely, if we would like to get an importance map that highlights the degradation of the model’s performance, we should add an inner step with gradient ascent within the loss. Hence by minimizing the resulting loss $-\ell(\theta + \epsilon \nabla_{\theta} \ell(\theta, x), x)$, we get the same formula for the gradients of the weights but with $\theta' = \theta + 2\epsilon \nabla_{\theta} \ell(\theta, x)$.

We hypothesize that applying meta-saliency to a saliency method should decrease correlation strength because allowing the network to update one SGD step in the direction of the min class should “destroy” the informativeness of the resulting saliency map. We use a learning rate $\epsilon = 0.001$ for the class sensitivity quantitative analysis. Figure 5 shows qualitatively that this appears to be the case: without meta-saliency, selective NormGrad and linear approximation yield max (class 2) and min class heatmaps that are highly positively and negatively correlated respectively. However, when meta-saliency is applied, the min class saliency map appears more random. Figure 7 shows results comparing the max-min class correlation scores with and

without meta-saliency. These results demonstrate that meta-saliency decreases max-min class correlation strength for nearly all saliency methods and suggest that meta-saliency can increase the class sensitivity for any saliency method.

4.5. Model weights sensitivity

[1] shows that some saliency methods (e.g., Guided Backprop in particular) are not sensitive to model weights as they are randomized in a cascading fashion from the end to the beginning of the network. Figure 8 shows qualitatively that, by the late conv layers, saliency maps for linear approximation and selective NormGrad are effectively scrambled (top two rows). It also highlights that, because meta-saliency increases class selectivity and is allowed to take one SGD in the direction of the target class, it takes relatively longer (i.e., more network depth) to randomize a meta-saliency heatmap (bottom row and see appendix).

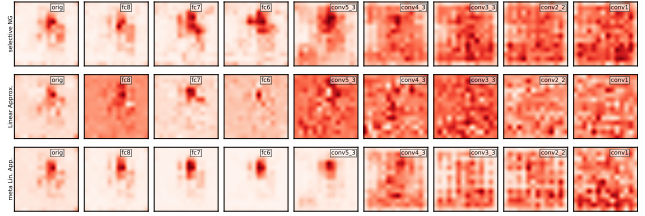


Figure 8. **Model weights sensitivity.** Sanity check by randomizing VGG16 model weights in a cascading fashion for the “Irish terrier” image from [1]. Top row: selective NormGrad, middle row: linear approximation, bottom row: linear approximation with meta-saliency (lower correlation with orig heatmap [leftmost col] is better). All methods look random after conv4_3. By comparing the last two rows at conv5_3, we see that meta-saliency enforces more class sensitivity than the non-meta variant.

5. Conclusions

We introduced a principled framework based on the contribution of each spatial location to the weights’ gradient. This framework unifies several existing backpropagation-based methods and allowed us to systematically explore the space of possible saliency methods. We use it for example to formulate NormGrad, a novel saliency method. We also studied how to combine saliency maps from different layers, discovering that it can consistently improve weak localization performance and produce high resolution maps. Finally, we introduced a class-sensitivity metric and proposed meta-saliency, a novel paradigm applicable to any existing method to improve sensitivity to the target class.

6. Acknowledgments

This work is supported by Mathworks/DTA, the Rhodes Trust (M.P.), EPSRC AIMS CDT and ERC 638009-IDIU.

References

- [1] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NeurIPS*, 2018. 1, 2, 7, 8
- [2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv*, 2016. 5
- [3] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proc. NIPS*, 2014. 1
- [4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 2015. 1, 2
- [5] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proc. NIPS*, 2017. 2
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 5
- [7] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015. 2, 5
- [8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017. 8
- [9] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proc. ICCV*, 2019. 2, 5
- [10] Ruth Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. CVPR*, 2017. 2
- [11] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv*, 2017. 1
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 5
- [13] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *arXiv*, 2018. 2
- [14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015. 3
- [15] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Xrai: Better attributions through regions. In *Proc. ICCV*, 2019. 2
- [16] 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. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 2019. 2
- [17] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv*, 2016. 1, 2, 3, 4
- [18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv*, 2018. 8
- [19] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. NIPS*, 2017. 2
- [20] Aravindh Mahendran and Andrea Vedaldi. Salient deconvolutional networks. In *Proc. ECCV*, 2016. 1, 2, 7
- [21] Leann Myers and Maria J Sirois. Spearman correlation coefficients, differences between. *Encyclopedia of statistical sciences*, 12, 2004. 5
- [22] Jose Oramas, Kaili Wang, and Tinne Tuytelaars. Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks. In *Proc. ICLR*, 2019. 2
- [23] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 1994. 8
- [24] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proc. BMVC*, 2018. 2
- [25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. KDD*, 2016. 2
- [26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 2
- [27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, 2017. 1, 2, 3, 4
- [28] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proc. ICML*, 2017. 2
- [29] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR*, 2014. 1, 2, 3, 4, 5
- [30] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *Proc. ICCV*, 2017. 2
- [31] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv*, 2017. 2
- [32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv*, 2014. 2
- [33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proc. ICML*, 2017. 2
- [34] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *Proc. CVPR*, 2017. 2
- [35] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proc. CVPR*, 2017. 2
- [36] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv*, 2019. 2
- [37] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, 2014. 1, 2

- [38] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *Proc. ECCV*, 2016. [1](#), [2](#), [5](#)
- [39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. CVPR*, 2016. [2](#), [4](#)