When Explanations Lie: Why Many Modified BP Attributions Fail

Leon Sixt [1] **Maximilian Granz** [1] **Tim Landgraf** [1]

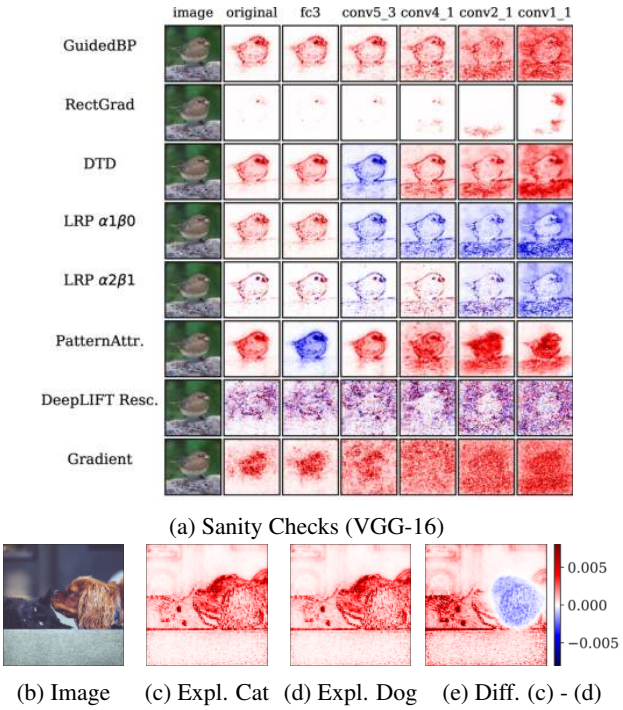
Abstract

Attribution methods aim to explain a neural network's prediction by highlighting the most relevant image areas. A popular approach is to backpropagate (BP) a custom relevance score using modified rules, rather than the gradient. We analyze an extensive set of modified BP methods: Deep Taylor Decomposition, Layer-wise Relevance Propagation, Excitation BP, PatternAttribution, DeepLIFT, Deconv, RectGrad, and Guided BP. We find empirically that the explanations of all mentioned methods, except for DeepLIFT, are independent of the parameters of later layers. We provide theoretical insights for this surprising behavior and also analyze why DeepLIFT does not suffer from this limitation. Empirically, we measure how information of later layers is ignored by using our new metric, cosine similarity convergence (CSC). The paper provides a framework to assess the faithfulness of new and existing modified BP methods theoretically and empirically.

1. Introduction

Explainable AI (XAI) aims to improve the interpretability of machine learning models. Different algorithms have been proposed to explain deep neural networks, but do the explanations reflect the inner workings correctly?

Saliency maps are a visual explanation of deep convolutional networks depicting relevant input areas for the prediction. For those explanations, (Adebayo et al., 2018) proposed a sanity check: if the parameters of the original model are randomized, the produced saliency map should change. Surprisingly, the saliency maps of GuidedBP (Springenberg et al., 2014) stay identical when the parameters of the last layer are altered (figure 1a). As the last layer predicts the final output, a method ignoring the last layer *does not* explain the network faithfully.



(a) Sanity Checks (VGG-16)

(b) Image (c) Expl. Cat (d) Expl. Dog (e) Diff. (c) - (d)

Figure 1: **(a)** Sanity Checks: Saliency maps should change if network parameters are randomized. Parameters are randomized from the last to the first layer. *Red* denotes positive and *blue* negative relevance. **(b-e)** Class insensitivity of $\text{LRP}_{\alpha 1 \beta 0}$ on VGG-16. Explanation for **(c)** *Persian cat (283)* and **(d)** *King Charles Spaniel (156)*. **(e)** Difference between (c) and (d) both normalized to $[0, 1]$. L1-norm of (e) $= 0.000371$.

We applied the sanity checks to an exhaustive set of modified BP methods: *Layer-wise Relevance Propagation (LRP)*, *Deep Taylor Decomposition (DTD)*, *PatternAttribution*, *Excitation BP*, *Deconv*, *GuidedBP*, *RectGrad*, and *DeepLIFT* (Bach et al., 2015; Montavon et al., 2017; Kindermans et al., 2018; Zhang et al., 2018; Zeiler & Fergus, 2014; Springenberg et al., 2014; Kim et al., 2019; Shrikumar et al., 2017). In addition to (Adebayo et al., 2018), which only reported GuidedBP to fail, we found that all mentioned methods, except for DeepLIFT, fail the sanity check and they therefore *do not* explain the predictions of deep neural networks faithfully.

[1]Dahlem Center of Machine Learning and Robotics, Freie Universität Berlin, Berlin, Germany. Correspondence to: Leon Sixt <leon.sixt@fu-berlin.de>.

Preprint

Modified BP methods estimate relevant areas by backpropagating a custom relevance score instead of the gradient. For example, DTD only backpropagates positive relevance. Modified BP methods are popular with practitioners (Yang et al., 2018; Sturm et al., 2016; Eitel et al., 2019). For example, (Schiller et al., 2019) improves the classification of whale sounds or (Böhle et al., 2019) explains MRT scans of Alzheimer patients using LRP$_{\alpha 1 \beta 0}$.

(Montavon et al., 2017; Bach et al., 2015; Kindermans et al., 2018) motivate the modification of the BP algorithm by analyzing linear models where the weight vector reflects the importance of each input directly. However, the methods fail the sanity check. Why does the motivation obtained form linear models not transfer to deep neural networks? What causes the explanations to become decoupled from the explained model?

Theoretically, we find that the z^+-rule - used by DTD, LRP$_{\alpha 1 \beta 0}$, and Excitation BP - yields a multiplication chain of non-negative matrices, which converge to a rank-1 matrix. A rank-1 matrix $C \in \mathbb{R}^{n \times m}$ can be written as an outer product $C = \boldsymbol{c}\boldsymbol{\gamma}^T$ where $\boldsymbol{c} \in \mathbb{R}^n$ and $\boldsymbol{\gamma} \in \mathbb{R}^m$. When C is multiplied with any vector $\boldsymbol{v}$, the direction of the resulting vector stays identical: $C\boldsymbol{v} = \boldsymbol{c}\boldsymbol{\gamma}^T\boldsymbol{v} = \lambda \boldsymbol{c}$ with $\lambda \in \mathbb{R}$. The scaling is irrelevant for saliency maps as they are normalized. If sufficiently converged, the backpropagated vector can merely switch the sign of the saliency map (see figure 1a) and saliency maps become visually identical for different classes (see figure 1b).

Empirically, we quantify the convergence to a rank-1 matrix using our novel cosine similarity convergence (CSC) metric. CSC allows to retrace, layer by layer, how modified BP methods lose information about previous layers. Using CSC, we observe that all analyzed modified BP methods, except for DeepLIFT, converge towards a rank-1 matrix on VGG-16 and ResNet-50. For sufficiently large values of α and β, LRP$_{\alpha\beta}$ does not converge but also produces rather noisy saliency maps.

2. Theoretical Analysis

2.1. Deep Neural Networks

Notation For our theoretical analysis, we consider feed forward neural networks with a ReLU activation function $[\boldsymbol{x}]^+ = \max(0, \boldsymbol{x})$. The neural network $f(\boldsymbol{x})$ contains n layers each with weight matrices W_l. The output of the l-th layer is denoted by $\boldsymbol{h}_l$. We use $[ij]$ to index the i, j element in W_l as in $W_{l_{[ij]}}$. To simplify notation, we absorb the bias terms into the weight matrix, and we omit the final softmax layer. We refer to the input with $\boldsymbol{h}_0 = x$ and to the output with $\boldsymbol{h}_n = f(x)$. The output of the l-th layer is given by:

$$\boldsymbol{h}_l = [W_l \boldsymbol{h}_{l-1}]^+ \tag{1}$$

All the results apply to convolutional neural networks as convolution can be expressed as matrix multiplication.

Gradient The gradient of the k-th output of the neural network w.r.t. the input $\boldsymbol{x}$ is given by:

$$\frac{\partial f_k(\boldsymbol{x})}{\partial \boldsymbol{x}} = W_1^T M_1 \frac{\partial f_k(\boldsymbol{x})}{\partial \boldsymbol{h}_1} = \prod_l^n \left(W_l^T M_l\right) \cdot \boldsymbol{v}_k, \tag{2}$$

where $M_l = \operatorname{diag}(\mathbb{1}_{h_l > 0})$ denotes the gradient mask of the ReLU operation. The last equality follows from recursive expansion. The vector $\boldsymbol{v}_k$ is a one-hot vector to select the k-th output.

The gradient of residual blocks also yield a product of matrices. The gradient of $\boldsymbol{h}_{l+1} = \boldsymbol{h}_l + g(\boldsymbol{h}_l)$ is given by:

$$\frac{\partial \boldsymbol{h}_{l+1}}{\partial \boldsymbol{h}_l} = I + G_{\partial g(\boldsymbol{h}_l)/\partial \boldsymbol{h}_l}, \tag{3}$$

where $G_{\partial g(\boldsymbol{h}_l)/\partial \boldsymbol{h}_l}$ denotes the derivation matrix of the residual block and I is the identity matrix.

The following methods modify the gradient definition and to distinguish the rules, we introduce the following notation: $r_l^{\nabla}(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{h}_l}$ which denotes the relevance at layer l for an input $\boldsymbol{x}$. For the gradient, the final saliency map is usually obtained by summing the absolute channel values of the relevance vector $r_0^{\nabla}(\boldsymbol{x})$ of the input layer.

2.2. Interpretability of Linear Models

The relevance of the input of a linear model can be calculated directly. Let $y = \boldsymbol{w}^T\boldsymbol{x}$ be a linear model with a single output scalar. The relevance of the input $\boldsymbol{x}$ to the output $\boldsymbol{y}_{[i]}$ is given by:

$$r_{\boldsymbol{x}}^{\text{Linear}}(\boldsymbol{x}) = \boldsymbol{w} \odot \boldsymbol{x}. \tag{4}$$

2.3. z^+-Rule

The z^+-rule is used by DTD (Montavon et al., 2017), Excitation BP (Zhang et al., 2018) and also corresponds to the LRP$_{\alpha 1 \beta 0}$ rule (Bach et al., 2015). The z^+-rule backpropagates positive relevance values, which are supposed to correspond to the positive evidence for the prediction. Let w_{ij} be an entry in the weight matrix W_l:

$$\begin{aligned} r_l^{z^+}(\boldsymbol{x}) &= Z_l^+ \cdot r_{l+1}^{z^+}(\boldsymbol{x}) \\ \text{where } {Z_l^+}^T &= \left(\frac{[w_{ij}\boldsymbol{h}_{l_{[j]}}]^+}{\sum_k [w_{ik}\boldsymbol{h}_{l_{[k]}}]^+} \right)_{[ij]}. \end{aligned} \tag{5}$$

Each entry in the derivation matrix Z_l^+ is obtained by measuring the positive contribution of the input neuron i to the output neuron j and normalizing by the total contributions

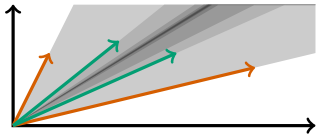


Figure 2: Carton of the cone formed by positive column vectors (organe). At each iteration, the cone shrinks.

to neuron j. The relevance from the previous layer $r_{l+1}^{z^+}$ is then distributed according to Z_l^+. The relevance function $r_l^{z^+} : \mathbb{R}^n \mapsto \mathbb{R}^m$ maps input $\boldsymbol{x}$ to a relevance vector of layer l. For the final layer the relevance is set to the value of the explained logit value, i.e. $r_n^{z^+}(\boldsymbol{x}) = f_k(\boldsymbol{x})$. Different to the vanilla backpropagation, algorithms using the z^+-rule do not apply a mask for the ReLU activation.

The relevance of multiple layers is computed by applying the z^+-rule to each of them. Similar as before with the gradient, we obtain a product of matrices, s: $C_k = \prod_l^k Z_l^+$. As the matrices are non-negative, the product converges to a rank-1 matrix. The the column vectors of a rank-1 matrix are linearly dependent $C = \boldsymbol{c}\boldsymbol{\gamma}^T$. Due to the convergence, the influence of later matirces decreases and when converged sufficently, the Z_{k+1}^+ matrix can only change the scaling as $CZ_{k+1}^+ = \boldsymbol{c}\boldsymbol{\gamma}^T Z_{k+1}^+ = \boldsymbol{c}\boldsymbol{\lambda}^T$ and for any vector $\boldsymbol{v}$: $CZ_{k+1}^+\boldsymbol{v} = \boldsymbol{c}\boldsymbol{\lambda}^T\boldsymbol{v} = t\boldsymbol{c}$ where $t \in \mathbb{R}$. For attribution methods, this means that the relevance vectors r_l of later layers $l > k$ do not contribute to the final result other than the scaling. However, the final decision of the network is made in the last layer and therefore a method converging to a rank-1 matrix *cannot* explain the network's true decision process. The following theorem states the conditions of convergence more precisely.

Theorem 1. *Let $A_1, A_2, A_3 \ldots$ be a sequence of non-negative matrices for which $\lim_{n\to\infty} A_n$ exists. We exclude the cases where one column of $\lim_{n\to\infty} A_n$ is the zero vector, a multiple of a standard basis vector or two columns are orthogonal to each other. Then the product of all terms of the sequence converges to a rank-1 matrix $\bar{C}$:*

$$\bar{C} := \prod_{i=1}^{\infty} A_i = \bar{\boldsymbol{c}}\boldsymbol{\gamma}^T . \tag{6}$$

In appendix A, we provide a rigorous proof of the theorem independent of matrix size. A similar result was given for squared matrices by (Hajnal, 1976).

The geometric intuition of the proof is depicted in figure 2. The column vectors of the first matrix are all non-negative and therefore in the positive quadrant. For the matrix multiplication A_iA_j, observe that $A_i\boldsymbol{a}_k$ is a non-negative linear combination of the column vectors of A_i , where $\boldsymbol{a}_k$ is the k-th column vector $A_{j_{[:k]}}$. The result will remain in the convex cone of the column vectors of A_i. The conditions stated in the theorem ensure that the cone shrinks with every iteration and it converges towards a single vector. In the appendix B, we simulate different matrix properties and find non-negative matrices to converge exponentially fast.

The Z^+ matrices of dense layers fulfill the conditions. Convolutions can also be written as matrix multiplications. For 1x1 convolutions, the kernels do not overlap and the columns corresponding to each location are orthogonal. In this case, the convergence happens only locally per feature map location. For convolutions with overlapping kernels, the global convergence is slower than for dense layers. In a ResNet-50 where the last convolutional stack has a size of (7x7), the overlapping of multiple (3x3) convolutions still induces a considerable global convergence (see LRP_{CMP} on ResNet-50).

2.4. Modified BP algorithms

LRP_z The LRP_z rule of Layer-wise Relevance Propagation modifies the back-propagation rule as follows:

$$r_l^{z-\text{LRP}}(\boldsymbol{x}) = Z_l \cdot r_{l+1}^{z-\text{LRP}}(\boldsymbol{x}),$$
$$\text{where } Z_l = \left(\frac{w_{ij}\boldsymbol{h}_{l_{[j]}}}{\sum_k w_{ik}\boldsymbol{h}_{l_{[k]}}} \right)^T_{[ij]} . \tag{7}$$

If only max-pooling, linear layers, and ReLU activations are used, it was shown that LRP_z corresponds to gradient$\odot$input, i.e. $r_0^{z-\text{LRP}}(\boldsymbol{x}) = \boldsymbol{x} \odot \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$ (Shrikumar et al., 2016; Kindermans et al., 2016; Ancona et al., 2017). LRP_z can be considered a gradient-based and not a modified BP method. The gradient is not converging to a rank-1 matrix and therefore gradient$\odot$input is also not converging.

$\text{LRP}_{\alpha\beta}$ separates the positive and negative influences:

$$r_l^{\alpha\beta}(\boldsymbol{x}) = \left(\alpha Z_l^+ - \beta Z_l^-\right) r_{l+1}^{\alpha\beta}(\boldsymbol{x}), \tag{8}$$

where Z_l^+ and Z_l^- correspond to the positive and negative entries of the matrix Z from LRP_z. α and β are dependent variables, as LRP conserves the total amount of relevance. The rules are $\alpha \geq 1$, and $\alpha - \beta = 1$. For $\text{LRP}_{\alpha 1 \beta 0}$, this rule corresponds to the z^+-rule, which converges. For $\alpha > 1$ and $\beta > 0$, the matrix $Z_l = \alpha Z_l^+ - \beta Z_l^-$ can contain negative entries. Our empirical results show that $\text{LRP}_{\alpha\beta}$ still converges for the most commonly used parameters $\alpha = 2, \beta = 1$ and even for a higher $\alpha = 5$ it converges on the ResNet-50.

Deep Taylor Decomposition uses the z^+-rule if the input to a convolutional or dense layer is in $[0, \infty]$, i.e. if the layers follow a ReLU activation. For inputs in $\mathbb{R}$, DTD also proposed the w^2-rule and the so-call $w^{\mathcal{B}}$ rule for bounded

inputs. Both rules were specifically designed to produce non-negative outputs. Theorem 1 applies and DTD converges to a rank-1 matrix necessarily.

PatternNet & PatternAttribution takes into account that the input $\boldsymbol{h}_l$ contains noise. If $\boldsymbol{d}_l$ corresponds to the noise and $\boldsymbol{s}_l$ to the signal, than $\boldsymbol{h}_l = \boldsymbol{s}_l + \boldsymbol{d}_l$. To assign the relevance towards the signal direction, the signal direction is estimated for each neuron from the layer's input $\boldsymbol{h}$ and the neuron's weight vector $\boldsymbol{w}_i = W_{[i:]}$:

$$\boldsymbol{a}_i = \frac{\operatorname{cov}[\boldsymbol{h}]\boldsymbol{w}_i}{\boldsymbol{w}_i^T \operatorname{cov}[\boldsymbol{h}]\boldsymbol{w}_i}. \tag{9}$$

PatternNet is designed to determine the relevant signal in the data. Let $A_{l[i:]} = \boldsymbol{a}_i$ be the corresponding signal matrix to the weight matrix W_l, the derivation rule for PatternNet is:

$$r_l^{\text{PN}}(\boldsymbol{x}) = A_l^T \cdot r_{l+1}^{\text{PN}}(\boldsymbol{x}), \tag{10}$$

PatternNet is also prone to converge to a rank-1 matrix. As PatternNet is not an attribution method but is designed to recover the relevant signal, it might be even desired to converge to a single direction, the signal direction.

The convergence of PatternNet follows from the computation of the pattern vectors $\boldsymbol{a}_i$ in equation 9. It is similar to a single step of the power iteration method $\boldsymbol{v}_{k+1} = C\boldsymbol{v}_k / \|C\boldsymbol{v}_k\|$. In appendix C, we provide details on the relationship to power iteration and also derive equation 9 from the equation given in (Kindermans et al., 2018). The power iteration method converges to the eigenvector with the largest eigenvalue exponentially fast.

All column vectors in $A_{[i:]} = \boldsymbol{a}_i$ underwent a single step of the power iteration and therefore tend to point towards the first eigenvector of $\operatorname{cov}[\boldsymbol{h}]$. This can also be verified empirically: the ratio of the first and second singular value $\sigma_1(A)/\sigma_2(A) > 6$ for almost all the VGG-16 patterns (see figure 3a), indicating a strong convergence of the matrix chain towards a single direction.

The findings from PatternNet are hard to transfer to PatternAttribution. The rule for PatternAttribution uses the Hadamard product of A_l and W_l:

$$r_l^{\text{PA}}(\boldsymbol{x}) = (W_l \odot A_l)^T \cdot r_{l+1}^{\text{PA}}(\boldsymbol{x}), \tag{11}$$

The Hadamard product complicates any analytic argument using properties of A or W. The theoretical results available (Ando et al., 1987; Zhan, 1997) did not allow us to show that PatternAttribution converges to a rank-1 matrix necessarily.

We provide a mix of theoretical and empirical insights on why it converges. The conditions of convergence can be studied well on the singular value decomposition: $M = USV$. Loosely speaking, the matrix chain will converge to a rank-1 matrix if the first σ_1 and second σ_2 singular values in S differ and if V_l and U_{l+1} are aligned such that higher singular values of S_l and S_{l+1} are multiplied together such that the ratio σ_1/σ_2 grows.

To see how well the per layer matrices align, we look at the inter-layer chain members: $M_l = \sqrt{S_l} V_l U_{l+1} \sqrt{S_{l+1}}$. In Figure 3, we display the ratio between the first and second singular values $\sigma_1(M_l)/\sigma_2(M_l)$. For $W \odot A$, the first singular value is considerably larger than for the plain weights W. Interestingly, the singular value ratio of inter-layer matrices shrinks for the plain W matrix. Whereas for PatternAttribution, the ratio even increases for some layers indicating that the Hadamard product leads to more alignment within the chain matrices.

DeepLIFT is the only tested modified BP method that does not converge to a rank-1 matrix. It can be seen as an extension of the backpropagation algorithm to work with finite differences:

$$\frac{f(\boldsymbol{x}) - f(\boldsymbol{x}^0)}{\boldsymbol{x} - \boldsymbol{x}^0} \tag{12}$$

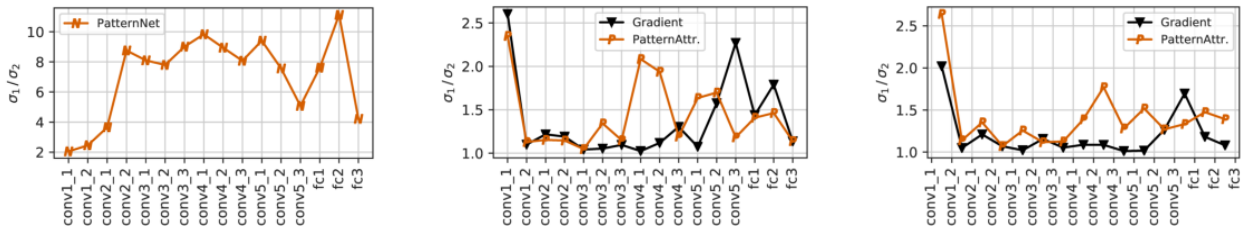
For the gradient, one would take the limit $\boldsymbol{x}^0 \to \boldsymbol{x}$. DeepLIFT uses a so-called reference point for $\boldsymbol{x}^0$ instead, such as a zeros or for images a blurred version of $\boldsymbol{x}$. The finite differences are backpropagated, similar to infinitesimal differences. The final relevance is the difference in the k-th logit: $r_l^{DL}(\boldsymbol{x}) = f_k(x) - f_k(x^0)$.

Additionally to the vanilla gradient, DeepLIFT separates positive and negative contributions. For ReLU activations, DeepLIFT uses either the RevealCancel or the Rescale rule. Please refer to (Shrikumar et al., 2017) for a description. The rule for linear layers is most interesting because it is the reason why DeepLIFT does not converge:

$$\begin{aligned} r_l^{DL+}(\boldsymbol{x}, \boldsymbol{x}^0) = & M_{>0}^T \odot \Big({W_l^+}^T r_{l+1}^{DL+}(\boldsymbol{x}, \boldsymbol{x}^0) \\ & + {W_l^-}^T r_{l+1}^{DL-}(\boldsymbol{x}, \boldsymbol{x}^0) \Big) \end{aligned} \tag{13}$$

where the mask $M_{>0}$ selects the weight rows corresponding to positive deltas ($0 < \Delta\boldsymbol{h}_l = \boldsymbol{h}_l - \boldsymbol{h}_l^0$). For negative relevance r_l^{DL-}, the rule is defined analogously. An interesting property of the linear rule is that negative and positive relevance can influence each other.

If the intermixing is removed by only considering W^+ for the positive rule and W^- for the negative rule, the two matrix chains become decoupled and converge. For the positive chain, this is clear. For the negative chain, observe that the multiplication of two non-positive matrices gives a non-negative matrix. Non-positive vectors $\boldsymbol{b}, \boldsymbol{c}$ cannot have an angle greater $90°$ and $\boldsymbol{c}^T\boldsymbol{b} = \|\boldsymbol{c}\| \|\boldsymbol{b}\| \cos(\boldsymbol{c}, \boldsymbol{b}) \geq 0$. In the evaluation, we included this variant as *DeepLIFT Ablation* and as predicted by the theory, it converges.



(a) PatternNet (b) $W = U_l S_l V_l$ (c) $\sqrt{S_l} V_l U_{l+1} \sqrt{S_{l+1}}$

Figure 3: PatternNet & PatternAttr.: **(a)(b)** Ratio between the first and second singular value σ_1/σ_2 for A_l, W_l, and $A_l \odot W_l$. **(c)** σ_1/σ_2 of inter-layer derivation matrices. For (b) (c), we sliced the 3x3 convolutional kernels to 1x1 kernels.

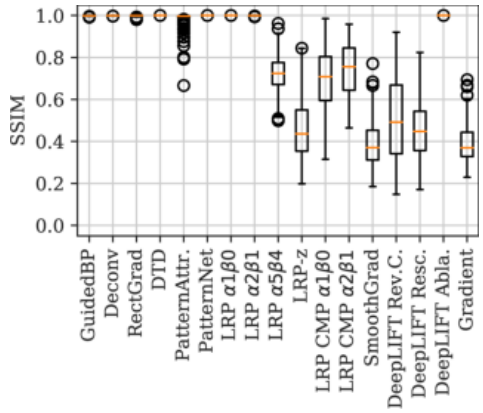


Figure 4: SSIM between saliency maps from the true and a random logit.

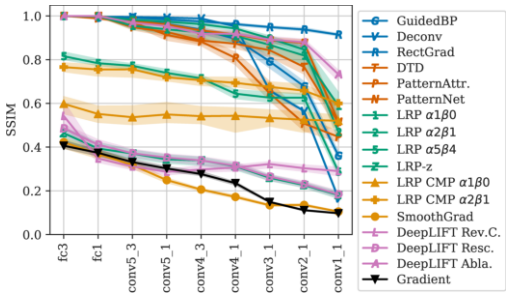


Figure 5: The parameters of the VGG-16 are randomized, starting from the last to the first layer. SSIM quantifies the difference to the saliency map from the original model. Intervals show 99% bootstrap confidences.

Guided BP & Deconv & RectGrad apply an additional ReLU to the gradient and it was shown to be invariant to the randomization of later layers previously in (Adebayo et al., 2018) and analyzed theoretically in (Nie et al., 2018):

$$r_l^{GBP}(\boldsymbol{x}) = W_l^T \left[M_l r_{l+1}^{GBP}(\boldsymbol{x}) \right]^+ . \tag{14}$$

$M_l = \text{diag}(\mathbb{1}_{h_1 > 0})$ denotes the gradient mask of the ReLU operation. For Deconv, the mask of the forward ReLU is omitted, and the gradients are rectified directly. RectGrad (Kim et al., 2019) is related to GuidedBP and set the lowest q percentile of the gradient to zero. As recommended in the paper, we used $q = 98$.

As a ReLU operation is applied to the gradient, the back-propagation is no longer a linear function. The ReLU also results in a different failure than before. (Nie et al., 2018) provides a theoretical analysis for GuidedBP and our results align with (Nie et al., 2018).

3. Evaluation

Setup We report results on a small network trained on CIFAR-10 (4x conv., 2x dense, see appendix D), a VGG-16 (Simonyan & Zisserman, 2014), and ResNet-50 (He et al., 2016). The last two are trained on the ImageNet dataset (Russakovsky et al., 2015), the standard dataset to evaluate attribution methods. The different networks cover different concepts: shallow vs. deep, forward vs. residual connections, multiple dense layers vs. a single one, using batch normalization or not. All results were computed on 200 images from the validation set. To justify the sample size, we show bootstrap confidence intervals in figure 5 (Efron, 1979). We used the implementation from the *innvestigate* and *deeplift* package (Alber et al., 2019; Shrikumar et al., 2017) and added support for residual connections. The experiments were run on a single machine with two graphic cards and take about a day to complete.

Random Logit We display the difference of saliency maps for the correct class logit and a random logit in figure 4. As the logit value is responsible for the predicted class, we would expect the saliency maps to change. We use the SSIM metric (Wang et al., 2004) to quantify the difference as in (Adebayo et al., 2018).

Sanity Check: Randomization of Parameters We follow (Adebayo et al., 2018) and randomized the parameters starting from the last layer to the first layer. For DTD and $\text{LRP}_{\alpha 1 \beta 0}$, the randomization of the last layer flips the sign

of the saliency map sometimes. We therefore compute the SSIM also between the inverted saliency map and report the maximum. In figure 5, we report the SSIM between the saliency maps (see also appendix G and figure 1a).[1]

Cosine Similarity Convergence Metric (CSC) Instead of randomizing the parameters, we randomize the backpropagated relevance vectors directly. We select layer k and set the corresponding relevance to $r_k(\boldsymbol{x}) := \boldsymbol{v}_1$ where $\boldsymbol{v}_1 \sim \mathcal{N}(0, I)$ and then backpropagate it as before. For example, for the gradient, we would do: $\frac{\partial h_k}{\partial h_1}\frac{\partial f(x)}{\partial h_k} := \frac{\partial h_k}{\partial h_1}\boldsymbol{v}_1$. We use the notation $r_l(\boldsymbol{x}|r_k{:=}\boldsymbol{v}_1)$ to describe the relevance r_l at layer l when the relevance of layer k is set to $\boldsymbol{v}_1$.

Using two random relevance vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \sim \mathcal{N}(0, I)$, we measure the convergence using the cosine similarity. If the relevance matrices converged to $C = \prod_l Z_l$, the columns of C are linear dependent $C = [\gamma_1 \boldsymbol{c}, \ldots, \gamma_k \boldsymbol{c}]$ and the result $C\boldsymbol{v} = \lambda \boldsymbol{c}$ is only a scaling of the column vector $\boldsymbol{c}$.

The backpropagated relevance vectors of $\boldsymbol{v}_1, \boldsymbol{v}_2$ will align more and more as the matrix chain converges. We quantify their alignment using the cosine similarity $s_{\cos}(r_l(\boldsymbol{x}|r_k{:=}\boldsymbol{v}_1), r_l(\boldsymbol{x}|r_k{:=}\boldsymbol{v}_2)))$ where $s_{\cos}(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b} \;/\; (\|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|)$.

If the relevance matrix chain converged, we have for both $\boldsymbol{v}_1, \boldsymbol{v}_2$: $r_l(\boldsymbol{x}|r_k{=}\boldsymbol{v}_i) = C\boldsymbol{v}_i = c\boldsymbol{\gamma}^T\boldsymbol{v}_i = \lambda_i \boldsymbol{c}$ where $\lambda_i = \boldsymbol{\gamma}^T\boldsymbol{v}_i$ and their cosine similarity will be one. The opposite direction is also true. If C has shape $n \times m$ with $n \leq m$ and if for n linear independent vectors $\boldsymbol{v}_i$, the cosine similarity $s_{\cos}(C\boldsymbol{v}_i, C\boldsymbol{v}_j) = 1$, then C is a rank-1 matrix.

An alternative way would have been to construct the derivation matrix $C_k = \prod_{l=1}^{k} Z_l$ and measure the ratio $\sigma_1(C_k)/\sigma_2(C_k)$ of the first to the second-largest singular value of C_k. Although this approach is well motivated theoretically, it has some performance downsides. C_k would be large and determining the singular values can be costly.

We use five different random vectors per sample – in total 1000 convergence paths. As the vectors are sampled randomly, it is unlikely to miss a region of non-convergence (Bergstra & Bengio, 2012).

For convolution layers, we compute the cosine similarity per feature map location, i.e. for a shape of (h, w, c), we obtain $h \cdot w$ values. The jump in cosine similarity for the input is a result of the input's low dimension of 3 channels. In figure 6, we plot the median cosine similarity for different networks and attribution methods (see appendix E for additional figures). We also report the histogram of the CSC at the first convolutional layer in figures 6e-6g. For more CSC figures, see appendix B.

[1] For GuidedBP, we report different saliency maps than shown in figure 2 of (Adebayo et al., 2018). We were able to confirm a bug in their implementation, resulting in saliency maps of GuidedBP and Guided-GradCAM to remain identical for early layers.

4. Results

Our random logit analysis reveals that converging methods produce almost identical saliency maps, independently of the output logit (SSIM very close to 1). The rest of the field (SSIM between 0.4 and 0.8) produces saliency maps different from the correct logit's map (see figure 4).

We observe the same distribution in the sanity check results (figure 11a). One group of methods produce similar saliency maps even when convolutional layers get randomized (SSIM close to 1). Again, the rest of the field is sensitive to parameter randomization. The same clustering can be observed for ResNet-50 (appendix F, figure 10).

Our CSC analysis confirms that random relevance vectors align throughout the backpropagation steps (figure 6). Except for LRP$_z$ and DeepLIFT, all methods show convergence up to at least 0.99 cosine similarity. LRP$_{\alpha 5 \beta 4}$ converges less strongly for VGG-16. Among the converging methods the rate of convergence varies. LRP$_{\alpha 1 \beta 0}$, PatternNet, the ablation of DeepLIFT converges fastest. PatternAttribution has a slower convergence rate – still exponential. For DeepLIFT Ablation, numerical instabilities result in a cosine similarity of 0 for the first layers of the ResNet-50. Even on the small 6-layer network, the median CSC is greater than 1-1e-6 for LRP$_{\alpha 1 \beta 0}$ (figure 6d).

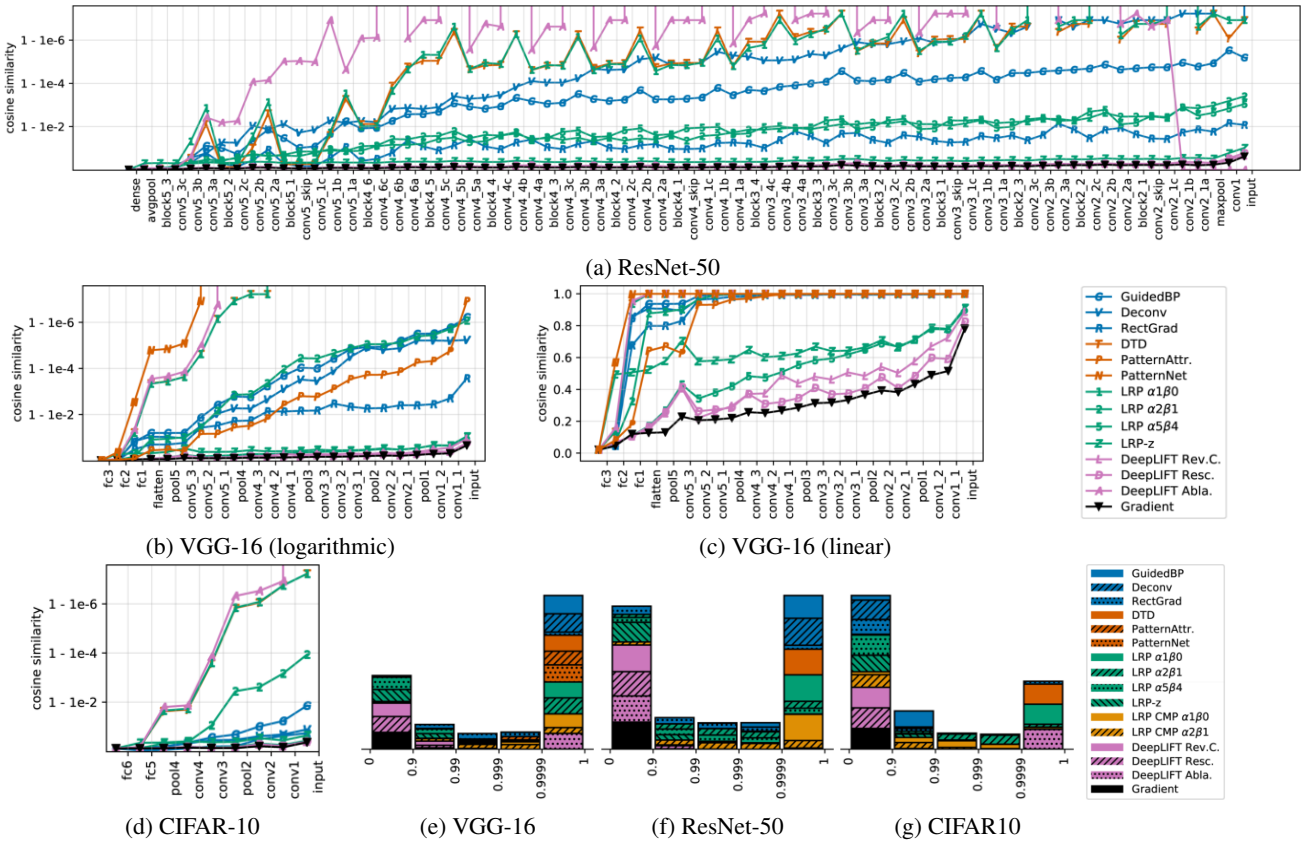
5. Discussion

When many modified BP methods do not explain the network faithfully, why was this not noticed before? First, it is easy to blame the neural network for unreasonable explanations – no ground truth exists. Second, MNIST, CIFAR, and ImageNet contain only a single object class per image – not revealing the class insensitivity. Finally, for some applications, it might not be too problematic if the saliency maps are independent of the later network's layers. For example, to explain Alzheimer's disease (Böhle et al., 2019), explaining local convolutional features might be sufficient.

When noticed, different ways to address the issue were proposed and an improved class sensitivity was reported (Kohlbrenner et al., 2019; Gu et al., 2019; Zhang et al., 2018). We find that the underlying convergence problem remains unchanged and discuss the methods below.

LRP$_{CMP}$ (Kohlbrenner et al., 2019; Lapuschkin et al., 2017) use LRP$_z$ for the final dense layers and LRP$_{\alpha\beta}$ for the convolutional layer. We report results for $\alpha = 1, 2$ as in (Kohlbrenner et al., 2019) in figure 7a.

For VGG-16, the saliency maps change when the net-



(a) ResNet-50

(b) VGG-16 (logarithmic)

(c) VGG-16 (linear)

(d) CIFAR-10

(e) VGG-16

(f) ResNet-50

(g) CIFAR10

Figure 6: **(a)-(d)** Median of the cosine similarity convergence (CSC) per layer between relevance vectors obtained from randomizing the relevance vectors of the final layer. **(e)-(g)** histogram of the distribution of the CSC after the first layer.

work parameters are randomized. However, structurally, the underlying image structure seems to be scaled only locally (see figure 7a). Inspecting the CSC path of the two LRP_{CMP} variants in figure 7c, we can see why. Both do not converge for the dense layers where LRP_{z} is used, but they converge as soon as $\text{LRP}_{\alpha\beta}$ is applied. The relevance vectors of the dense layer can change the coarse local scaling, but they cannot alter the direction of the relevance vectors of earlier layers to highlight different details.

In the backward-pass of the ResNet-50, the global-averaging layer assigns the identical gradient vector to each location of the last conv. layer. Furthermore, the later conv. layers operate on (7x7), where even a few 3x3 convolutions have a dense FoV. LRP_{CMP} does not resolve the global convergence for the ResNet-50.

Contrastive LRP (Gu et al., 2019) noted the lack of class sensitivity and proposed to increase it by subtracting two saliency maps. The first saliency map explains only the logit $\boldsymbol{y}_k = \boldsymbol{y} \odot \boldsymbol{m}_k$, where $\boldsymbol{m}_k$ is a one-hot vector and the second explains the opposite $\boldsymbol{y}_{\neg k} = \boldsymbol{y} \odot (1 - \boldsymbol{m}_k)$:

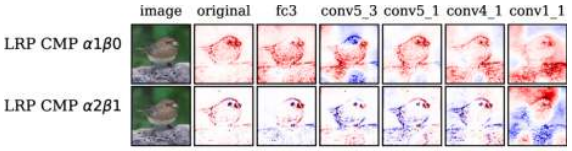
$$\begin{aligned} &\max(0, n(r_{\boldsymbol{x}}^{z^+}(\boldsymbol{x}|r_{\text{logits}} := \boldsymbol{y}_k)) \\ &\quad - n(r_{\boldsymbol{x}}^{z^+}(\boldsymbol{x}|r_{\text{logits}} := \boldsymbol{y}_{\neg k})) \end{aligned} \tag{15}$$

$n(.)$ normalizes each saliency map by its sum. The results of Contrastive LRP are similar to figure 1e, no $\max$ is applied. The underlying convergence problem is not resolved.

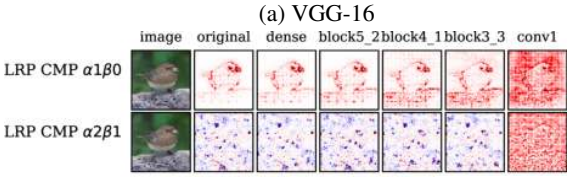
Contrastive Excitation BP The lack of class sensitivity of the z^+-rule was noted in (Zhang et al., 2018) and to increase it, they proposed to change the backpropagation rule of the final fully-connected layer to:

$$r^{\text{cEBP}}_{\text{final fc}}(x) = (Z^+_{\text{final fc}} - N^+_{\text{final fc}})\boldsymbol{m}_k, \tag{16}$$

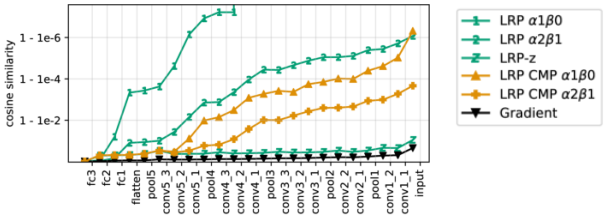
where $\boldsymbol{m}_k$ is a one-hot vector selecting the explained class. The added $N^+_{\text{final fc}}$ is computed as the $Z^+_{\text{final fc}}$ but on the negative weights $-W_{\text{final fc}}$. Note that the combination of the two matrices introduces negative entries. Class sensitivity is increased. It does also not resolve the underlying convergence problem. If, for example, more fully-connected layers would be used, the saliency maps would become globally class insensitive again.



(a) VGG-16



(b) ResNet-50



(c) VGG-16

Figure 7: **(a-b)** Sanity checks and **(c)** CSC for LRP$_{CMP}$.

Texture vs. Contours (Geirhos et al., 2018) found that deep convolutional networks are more sensitive towards texture and not the shape of the object. For example, the shape of a cat filled with an elephant texture will be wrongly classified as an elephant. However, modified BP methods highlight the contours of objects rather.

Not converging Attribution Methods Besides modified BP attribution methods, there also exist gradient averaging and black-box methods. SmoothGrad (Smilkov et al., 2017) and Integrated Gradients (Sundararajan et al., 2017) average the gradient. *CAM* and *Grad-CAM* (Zhou et al., 2016; Selvaraju et al., 2017) determine important areas by the activation of the last convolutional layer. Black-box attribution methods only modify the input of the model but do not rely on the gradient or other model internals. The most prominent black-box methods are *Occlusion*, *LIME*, *SHAP* (Zeiler & Fergus, 2014; Ribeiro et al., 2016; Lundberg & Lee, 2017). For example, *Occlusion* measures the sensitivity of the network prediction by masking out image patches. (Schulz et al., 2020) applies an information bottleneck to remove unimportant information. TCAV (Kim et al., 2018) explains models using more higher-level concepts. All here mentioned attribution methods *do not* converge, as they either rely on the gradient or treat the model as black-box. Only when the BP algorithm is modified, the convergence problem can occur. The here mentioned algorithms might still suffer from other limitations.

Limitations Also, we tried to include most modified BP attribution methods, some are not covered by our evaluation (Nam et al., 2019; Wang et al., 2019; Huber et al., 2019). In our theoretical analysis of PatternAttribution, our argument why it converges is partially based on empirical observations performed on a single set of pattern matrices.

6. Related Work

The limitations of explanation methods were studied before. (Viering et al., 2019) alter the explanations of Grad-CAM arbitrarily by modifying the model architecture only slightly. Similarly, (Slack et al., 2020) construct a biased classifier that can hide its biases from LIME and SHAP. The theoretic analysis (Nie et al., 2018) indicates that GuidedBP tends rather reconstruct the input then to explain the network's decision. (Adebayo et al., 2018) showed GuidedBP to be independent of later layers' parameters.

(Kindermans et al., 2018) show that LRP, GuidedBP and Deconv produce incorrect explanations for linear models if the input contains noise. (Zhang et al., 2018; Gu et al., 2019; Kohlbrenner et al., 2019; Montavon et al., 2019; Tsunakawa et al., 2019) noted class insensitivity of different modified BP methods, but they rather proposed ways to improve the class sensitivity than to provide correct reasons why modified BP methods are class insensitive. Other than argued in (Gu et al., 2019), the class insensitivity is not caused by missing ReLU masks and Pooling switches.

A different approach for testing attribution methods is human subject studies (Alqaraawi et al., 2020; Doshi-Velez & Kim, 2017; Lage et al., 2018).

Our CSC measure has some similarities with the work (Balduzzi et al., 2017), which analyzes the effect of skip connections on the gradient. They measure the convergence between the gradient vector from different samples using the effective rank (Vershynin, 2012).

To our best knowledge, we are the first to identify the reason why many modified BP methods do not explain the decision of deep neural networks faithfully.

7. Conclusion

In our paper, we analyzed modified BP methods which aim to explain the predictions of deep neural networks. Our analysis revealed that these attribution methods themselves are little understood and have theoretical properties contrary to their goal. Of all analyzed modified BP algorithms, we found that only DeepLIFT as a finite backpropagation algorithm is theoretically well-founded. For attribution methods, a sound theoretical motivation is important – especially as ground truth is so hard to get.

8. Acknowledgements

We thank Benjamin Wild and David Dormagen for stimulating discussions. We also thank Avanti Shrikumar for answering our questions and helping us with the DeepLIFT implementation. The comments by Agathe Balayn, Karl Schulz, and Julian Stastny improved the manuscript. A special thanks to the anonymous reviewer 1 of our paper (Schulz et al., 2020), who encouraged us to report results on the sanity checks – the starting point of this paper. LS was supported by the Elsa-Neumann-Scholarship by the state of Berlin. We are also grateful to Nvidia for providing us with a Titan Xp and to ZEDAT for granting us access to their HPC system.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pp. 9505–9515, 2018.

Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., Samek, W., Müller, K.-R., Dähne, S., and Kindermans, P.-J. innvestigate neural networks! *Journal of Machine Learning Research*, 20(93):1–8, 2019.

Alqaraawi, A., Schuessler, M., Weiß, P., Costanza, E., and Berthouze, N. Evaluating saliency map explanations for convolutional neural networks: A user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, IUI '20, pp. 263–274, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3377325.3377519.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. A unified view of gradient-based attribution methods for deep neural networks. In *NIPS 2017-Workshop on Interpreting, Explaining and Visualizing Deep Learning*. ETH Zurich, 2017.

Ando, T., Horn, R. A., and Johnson, C. R. The singular values of a hadamard product: a basic inequality. *Linear and Multilinear Algebra*, 21(4):345–365, 1987. doi: 10.1080/03081088708817810.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 2015. doi: 10.1371/journal.pone.0130140.

Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K. W.-D., and McWilliams, B. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 342–350. JMLR.org, 2017.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Böhle, M., Eitel, F., Weygandt, M., and Ritter, K. Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer's disease classification. *Frontiers in Aging Neuroscience*, 11:194, 2019. ISSN 1663-4365. doi: 10.3389/fnagi.2019.00194.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. ISSN 00905364.

Eitel, F., Soehler, E., Bellmann-Strobl, J., Brandt, A. U., Ruprecht, K., Giess, R. M., Kuchling, J., Asseyer, S., Weygandt, M., Haynes, J.-D., Scheel, M., Paul, F., and Ritter, K. Uncovering convolutional neural network decisions for diagnosing multiple sclerosis on conventional mri using layer-wise relevance propagation. *NeuroImage: Clinical*, 24:102003, 2019. ISSN 2213-1582. doi: https://doi.org/10.1016/j.nicl.2019.102003.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *ArXiv*, abs/1811.12231, 2018.

Gu, J., Yang, Y., and Tresp, V. Understanding Individual Decisions of CNNs via Contrastive Backpropagation. *arXiv:1812.02100 [cs]*, September 2019.

Hajnal, J. On products of non-negative matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 79(3):521–530, May 1976. ISSN 0305-0041, 1469-8064.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huber, T., Schiller, D., and André, E. Enhancing explainability of deep reinforcement learning through selective layer-wise relevance propagation. In Benzmüller, C. and Stuckenschmidt, H. (eds.), *KI 2019: Advances in Artificial Intelligence*, pp. 188–202, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30179-8.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pp. 2673–2682, 2018.

Kim, B., Seo, J., Jeon, S., Koo, J., Choe, J., and Jeon, T. Why are Saliency Maps Noisy? Cause of and Solution to Noisy Saliency Maps. *arXiv e-prints*, 2019.

Kindermans, P.-J., Schütt, K., Müller, K.-R., and Dähne, S. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016.

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., Erhan, D., Kim, B., and Dähne, S. Learning how to explain neural networks: Patternnet and patternattribution. In *International Conference on Learning Representations*, 2018.

Kohlbrenner, M., Bauer, A., Nakajima, S., Binder, A., Samek, W., and Lapuschkin, S. Towards best practice in explaining neural network decisions with lrp, 2019.

Lage, I., Ross, A., Gershman, S. J., Kim, B., and Doshi-Velez, F. Human-in-the-loop interpretability prior. In *Advances in Neural Information Processing Systems*, pp. 10159–10168, 2018.

Lapuschkin, S., Binder, A., Muller, K.-R., and Samek, W. Understanding and comparing deep neural networks for age and gender classification. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Montavon, G., Binder, A., Lapuschkin, S., Samek, W., and Müller, K.-R. Layer-wise relevance propagation: an overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 193–209. Springer, 2019.

Nam, W.-J., Gur, S., Choi, J., Wolf, L., and Lee, S.-W. Relative attributing propagation: Interpreting the comparative contributions of individual units in deep neural networks. *arXiv:1904.00605 [cs]*, Nov 2019.

Nie, W., Zhang, Y., and Patel, A. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *International Conference on Machine Learning*, pp. 3806–3815, 2018.

Ribeiro, M. T., Singh, S., and Guestrin, C. "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*, KDD '16, pp. 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939778.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Schiller, D., Huber, T., Lingenfelser, F., Dietz, M., Seiderer, A., and André, E. Relevance-Based Feature Masking: Improving Neural Network Based Whale Classification Through Explainable Artificial Intelligence. In *Proc. Interspeech 2019*, pp. 2423–2427, 2019.

Schulz, K., Sixt, L., Tombari, F., and Landgraf, T. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Shrikumar, A., Greenside, P., Shcherbina, A., and Kundaje, A. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153. JMLR.org, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pp. 180–186, New York, NY, USA, 2020. Association for

Computing Machinery. ISBN 9781450371100. doi: 10.1145/3375627.3375830.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv:1706.03825 [cs, stat]*, 2017.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *arXiv e-prints*, 2014.

Sturm, I., Lapuschkin, S., Samek, W., and Müller, K.-R. Interpretable deep neural networks for single-trial eeg classification. *Journal of neuroscience methods*, 274: 141–145, 2016.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017.

Tsunakawa, H., Kameya, Y., Lee, H., Shinya, Y., and Mitsumoto, N. Contrastive relevance propagation for interpreting predictions by a single-shot object detector. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, July 2019. doi: 10.1109/IJCNN.2019.8851770.

Vershynin, R. *Introduction to the non-asymptotic analysis of random matrices*, pp. 210–268. Cambridge University Press, 2012. doi: 10.1017/CBO9780511794308.006.

Viering, T., Wang, Z., Loog, M., and Eisemann, E. How to manipulate cnns to make them lie: the gradcam case, 2019.

Wang, S., Zhou, T., and Bilmes, J. Bias also matters: Bias attribution for deep neural network explanation. In *International Conference on Machine Learning*, pp. 6659–6667, 2019.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Yang, Y., Tresp, V., Wunderle, M., and Fasching, P. A. Explaining therapy predictions with layer-wise relevance propagation in neural networks. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 152–162, June 2018.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhan, X. Inequalities for the Singular Values of Hadamard Products. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1093–1095, October 1997. ISSN 0895-4798, 1095-7162. doi: 10.1137/S0895479896309645.

Zhang, J., Bargal, S. A., Lin, Z., Brandt, J., Shen, X., and Sclaroff, S. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921–2929, 2016.

A. Proof of Theorem 1

We give a general proof under which conditions non-negative matrices converge. First, we outline the conditions on the matrix sequence A_n. Then, we state the theorem again and sketch our proof to give the reader a better overview. Finally, we prove the theorem in 5 steps.

Conditions on A_n The first obvious condition is that the $(A_n)_{n\in\mathbb{N}}$ is a sequence of non-negative matrices such that A_i, A_{i+1} have the correct size to be multiplied together. Secondly, as we calculate angles between column vector in our proof, no column of A_n should be zero. The angle between a zero vector and any other vector is undefined. Finally, the size of A_n should not increase infinitely, i.e. an upper bound on the size of the A_i's exists such that $A_i \in \mathbb{R}^{m\times l}$ where $m, l \leq L$ for some $L \in \mathbb{N}$.

Definition 1. We say $\lim_{n\to\infty} A_n$ **exists**, if for all $m, l \in \mathbb{N}$ the subsequences $A_{n_{k(m,l)}} \in \mathbb{R}^{m\times l}$ that consist of all terms that have size $m \times l$ converge elementwise. Note that even if there only finitely many, say A_N is the last term with $A_n \in \mathbb{R}^{m\times l}$, we say $\lim_{k\to\infty} A_{n_{k(l,m)}} = A_N$.

Theorem 1. *Let $A_1, A_2, A_3 \dots$ be a sequence of non-negative matrices as described above such that $\lim_{n\to\infty} A_n$ exists. We exclude the cases where one column of $\lim_{n\to\infty} A_n$ is the zero vector, a multiple of a standard basis vector or two columns are orthogonal to each other. Then the product of all terms of the sequence converges to a rank-1 matrix $\bar{C}$:*

$$\bar{C} := \prod_{i=1}^{\infty} A_i = \bar{\boldsymbol{c}}\boldsymbol{\gamma}^T . \tag{17}$$

To clarify things, this is the specific way how $\lim_{n\to\infty} A_n$ should *not* look like:

$$\underbrace{\begin{pmatrix} \boldsymbol{v}_1 & \dots & \boldsymbol{v}_l & \boldsymbol{v}_{l+1} & \dots & \boldsymbol{v}_m & \boldsymbol{v}_{m+1} & \dots & \boldsymbol{v}_n \end{pmatrix}}_{\text{Theory}} \quad \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}}_{\text{Example}}$$

$\boldsymbol{v}_i = \lambda \boldsymbol{e}_{j_i}$ | orthogonal | $\boldsymbol{v}_i = 0$

up to ordering of the columns.

Proof sketch To show that $\prod_i^{\infty} A_i$ converges to a rank-1 matrix, we do the following steps:

(1) We define a sequence s_n as the cosine of the maximum angle between the column vectors of $M_n := \prod_{i=1}^{n} A_i$.

(2) We show that the sequence s_n is monotonic and bounded and therefore converging.

(3) We assume $\lim_{n\to\infty} s_n \neq 1$ and analyze two cases where we do not get a contradiction. Each case yields an equation on $\lim_{n\to\infty} A_n$.

(4) In both cases, we find lower bounds on s_n: $\alpha_n s_{n-1}$ and $\alpha'_n s_{n-1}$ that are becoming infinitely large, unless we have $\lim_{n\to\infty} \alpha_n = 1$ (case 1) or $\lim_{n\to\infty} \alpha'_n = 1$ (case 2).

(5) The lower bounds lead to equations on $\lim_{n\to\infty} A_n$ for non-convergence. The only solutions, we obtain for $\lim_{n\to\infty} A_n$, are those explicitly excluded in the theorem. We still get a contradiction and $\lim_{n\to\infty} s_n = 1$.

Proof **(1)** Let $M_n := \prod_{i=1}^{n} A_i$ be the product of the matrices $A_1 \cdot \ldots \cdot A_n$. We define a sequence on the angles of column vectors of M_n using the cosine similarity. Let $\boldsymbol{v}_1(n), ..., \boldsymbol{v}_{k(n)}(n)$ be the column vectors of M_n. Note, the angles are well defined between the columns of M_n. The columns of M_n cannot be a zero vector as we required A_n to have no zero columns. Let s_n is the cosine of the maximal angle between the columns of M_n:

$$s_n = \min_{i,j} \mathrm{s}_{\cos}(\boldsymbol{v}_i(n), \boldsymbol{v}_j(n)) := \min_{i,j} \frac{\langle \boldsymbol{v}_i(n), \boldsymbol{v}_j(n)\rangle}{\|\boldsymbol{v}_i(n)\|\|\boldsymbol{v}_j(n)\|}, \tag{18}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product. We show that the maximal angle converges to 0 as $\lim_{n\to\infty} s_n = 1$, which is equivalent to M_n converging to a rank-1 matrix. In the following, we take a look at two consecutive elements of the sequence s_n and check by how much the sequence increases.

(2) We show that the sequence s_n is monotonic and bounded and therefore converging. Assume $\boldsymbol{a}_{n+1}$ and $\boldsymbol{b}_{n+1}$ are the two columns of A_{n+1} which produce the columns m_1 and m_2 of M_{n+1} with the maximum angle:

$$s_{n+1} = \mathrm{s}_{\cos}(m_1, m_2) = \mathrm{s}_{\cos}(M_n \boldsymbol{a}_{n+1}, M_n \boldsymbol{b}_{n+1}). \tag{19}$$

We also assume that $\|\boldsymbol{v}_i(n)\| = 1$ for all i, since the angle is independent of length. To declutter notation, write $\boldsymbol{v}_i(n) =: \boldsymbol{v}_i$, $\boldsymbol{a}_n = \boldsymbol{a} = (a_1, ..., a_k)^T$ and $\boldsymbol{b}_n = \boldsymbol{b} = (b_1, ..., b_k)^T$. We now show that s_n is monotonic and use the definition of the cosine similarity:

$$s_{n+1} = \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{\| \sum_i a_i \boldsymbol{v}_i \| \| \sum_i b_i \boldsymbol{v}_i \|} \tag{20}$$

Using the triangle inequality $\| \sum a_i \boldsymbol{v}_i \| \leq \sum a_i \| \boldsymbol{v}_i \|$ we get:

$$s_{n+1} \geq \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{(\sum_i a_i \|\boldsymbol{v}_i\|)(\sum_i b_i \|\boldsymbol{v}_i\|)} \tag{21}$$

As we assumed that the $\|\boldsymbol{v}_i\| = 1$, we know that $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = s_{\cos}(\boldsymbol{v}_i, \boldsymbol{v}_j)$ which must be greater than the smallest cosine similarity s_n:

$$s_{n+1} \geq \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{(\sum_i a_i)(\sum_i b_i)} \geq \frac{\sum_{ij} a_i b_j}{(\sum_i a_i)(\sum_i b_i)} s_n = s_n \tag{22}$$

Therefore s_n is monotonically increasing and upper-bounded by 1 as the cosine. Due to the monotone convergence theorem, it will converge. The rest of the proof investigates if the sequence s_n converges to 1 and if so, under which conditions.

(3) We look at two consecutive sequence elements and measure the factor α by which they increase: $s_{n+1} \geq \alpha s_n$. We are using proof by contradiction and assume that s_n does not converge to 1. We get two cases, each with a lower bound on the factor of increase. For both cases, we find a lower bound for $\alpha > 1$ for all n which would mean that s_n is diverging to ∞ – a contradiction. Under certain conditions on $\lim_{n\to\infty} A_n$, we do not find a lower bound $\alpha > 1$. We find that these conditions correspond to the conditions explicitly excluded in the theorem and therefore M_n converges to a rank-1 matrix.

Case 1: Let $t_n := \langle \boldsymbol{v}_l(n), \boldsymbol{v}_m(n) \rangle$ ($l \neq m$) and assume that there exists a subsequence t_{n_k} of t_n that does not converge to $\lim_{n\to\infty} s_n$. So there is an $\varepsilon > 0$ such that $\langle \boldsymbol{v}_l(n_k), \boldsymbol{v}_m(n_k) \rangle \geq (1+\varepsilon) s_{n_k}$ for all $k \geq K$ for some $K \in \mathbb{N}$.

We multiply equation 22 by 1 and get:

$$s_{n+1} \geq \frac{\sum a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{(\sum_i a_i)(\sum_i b_i)} = \frac{\sum a_i b_j \frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{s_n}}{(\sum_i a_i)(\sum_i b_i)} s_n, \tag{23}$$

We will now pull terms corresponding to the pair (l, m) out of the sum and for all terms in the sum, we lower bound $\frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{s_n} \geq 1$ by one. Let the index $I := \{(i,j) \mid (i,j) \neq (l,m), (m,l)\}$ include all other terms:

$$s_{n+1} \geq \frac{\sum_I a_i b_j + (a_l b_m + a_m b_l) \frac{\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{s_n}}{(\sum_i a_i)(\sum_i b_i)} s_n \tag{24}$$

We know that $\langle \boldsymbol{v}_l(n_k), \boldsymbol{v}_m(n_k) \rangle \geq (1+\varepsilon) s_{n_k}$:

$$s_{n_{k+1}} \geq s_{n_k+1} \geq \frac{\sum_I a_i b_j + (a_l b_m + a_m b_l)(1+\varepsilon)}{(\sum_i a_i)(\sum_i b_i)} s_{n_k} \tag{25}$$

We absorb the m, l factors back into the sum:

$$s_{n_{k+1}} \geq \frac{\sum a_i b_j + \overbrace{(a_l b_m + a_m b_l)}^{=:r_{n_k}} \varepsilon}{(\sum_i a_i)(\sum_i b_i)} s_{n_k} = \left(1 + \frac{r_{n_k} \varepsilon}{(\sum_i a_i)(\sum_i b_i)}\right) s_{n_k} \geq \underbrace{\left(1 + \frac{r_{n_k} \varepsilon}{\bar{q}}\right)}_{=:\alpha_{n_k}} s_{n_k} \tag{26}$$

where $\bar{q}$ is an upper bound on $\sum_{ij} a_i b_j$ which exists since $\lim_{n\to\infty} A_n$ exists, which is also why $\lim_{n\to\infty} r_{n_k}$ exists.

(4) Case 1: Define $r_n = a_l(n)b_m(n) + a_m(n)b_l(n)$ analogous to r_{n_k}. So if $\lim_{n\to\infty} r_n = \lim_{k\to\infty} r_{n_k} \neq 0$, the factor by which s_n increases would be greater that one by a constant – a contradiction:

$$s_{n_k} \geq (1+c)^{n'} s_{n_1} > 0 \tag{27}$$

where $n_k - n'$ is the number of cases where $r_{n_k} = 0$ and $c > 0$ is a lower bound on the set $\{r_{n_k} \neq 0\}$. As we assumed $\lim_{n\to\infty} r_n \neq 0$, $c > 0$ for an infinite number of cases and therefore $n' \to \infty$ when $k \to \infty$.

To end case 1, we have to ensure that the first sequence element is greater than zero: $s_{n_1} \geq s_1 > 0$. This is not the case if the first N matrices have two orthogonal columns, $s_1 = ... = s_N = 0$. We can then skip the first N matrices and define s_1 on A_{N+1} (set $A_i = A_{N+i}$). We know N has to be finite, as $\lim_{n\to\infty} A_n$ has no two columns that are orthogonal and therefore A_{N+1} has no orthogonal columns.

(3) Case 2: No subsequence of t_n, as defined in case 1, exists , i.e. for all $\varepsilon > 0$ no subsequence $t_{n_k} = \langle \boldsymbol{v}_l(n), \boldsymbol{v}_m(n) \rangle \geq (1+\varepsilon) s_{n_k}$ exists. Then t_n and s_n converge to the same value: $\lim_{n\to\infty} t_n - s_n = 0$. Since we assumed that s_n does not converge to one, it must converge to a value smaller than 1 by a constant ε'. An $N \in \mathbb{N}$ exists such that for all $n \geq N$ there is an $\varepsilon' > 0$ with $\langle \boldsymbol{v}_l(n), \boldsymbol{v}_m(n) \rangle \leq 1 - \varepsilon'$. We derive a second lower bound:

$$s_{n+1} = \frac{\sum_{ij} a_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{\| \sum_i a_i \boldsymbol{v}_i \| \| \sum_i b_i \boldsymbol{v}_i \|} \geq \frac{\sum_{ij} a_i b_j}{\| \sum_i a_i \boldsymbol{v}_i \| \| \sum_i b_i \boldsymbol{v}_i \|} s_n \tag{28}$$

where we used the fact that $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle \geq s_n$ .We now try to find a lower bound for the square of this factor similar as done before:

$$\frac{(\sum a_i b_j)^2}{\| \sum a_i \boldsymbol{v}_i \|^2 \| \sum b_i \boldsymbol{v}_i \|^2} = \frac{(\sum a_i b_j)^2}{(\sum a_i a_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle)(\sum b_i b_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle)} \tag{29}$$

$$= \frac{(\sum a_i b_j)^2}{(\sum_I a_i a_j + 2a_l a_m \langle \boldsymbol{v}_l, \boldsymbol{v}_m \rangle)(\sum_I b_i b_j + 2b_l b_m \langle \boldsymbol{v}_l, \boldsymbol{v}_m \rangle)} \tag{30}$$

$$\geq \frac{(\sum a_i b_j)^2}{(\sum_I a_i a_j + 2a_l a_m (1-\varepsilon))(\sum_I b_i b_j + 2b_l b_m (1-\varepsilon))} \tag{31}$$

$$= \frac{q^2}{q^2 - \varepsilon(\sum 2a_l a_m b_i b_j + \sum 2b_l b_m a_i a_j - 4\varepsilon a_l a_m b_l b_m)} \tag{32}$$

$$\geq \frac{q^2}{q^2 - \varepsilon^2 \underbrace{(\sum 2a_l a_m b_i b_j + \sum 2b_l b_m a_i a_j - 4a_l a_m b_l b_m)}_{=: r'_n}} \tag{33}$$

$$= \frac{q^2}{q^2 - \varepsilon^2 r'_n} = 1 + \frac{\varepsilon^2 r'_n}{q^2 - \varepsilon^2 r'_n} \geq 1 + \frac{\varepsilon^2 r'_n}{\bar{q}^2 - \varepsilon^2 r'_n} =: \alpha'^2_n \tag{34}$$

where $q^2 = (\sum a_i b_j)^2$ and $\bar{q}^2$ is an upper bound on q^2 for all n. Note that $q^2 - \varepsilon^2 r'_n > 0$, since q^2 has all terms that r'_n has but more.

(4) Case 2: So r'_n is a sequence that converges to zero. Otherwise, the factor by which s_n increases would be greater than one by at least a constant for infinitely many n. As in the previous case 1, this would lead to a contradiction.

(5) Case 1 and case 2 are complements from which we obtain two equations for $\lim_{n\to\infty} A_n$. Let $\boldsymbol{a}_k = (a_1, ..., a_k)^T$ and $\boldsymbol{b}_k = (b_1, ..., b_k)^T$ be columns of $\lim_{n\to\infty} A_n$. We get one equation per case. For all (i, j) with $i < j$ we have:

$$\textbf{Case 1:} \lim_{n\to\infty} r_n = 0 \Rightarrow a_i b_j = a_j b_i = 0 \quad \text{or} \quad \textbf{Case 2:} \lim_{n\to\infty} r'_n = 0 \Rightarrow a_i a_j = b_i b_j = 0, \tag{35}$$

where the first equation comes from $\lim_{n\to\infty} r_n = 0$ and the second from $\lim_{n\to\infty} r'_n = 0$.

For equation 35 to be true, the following set of equations have to hold:

$$S(k) := \{\forall i = 1, ..., k : a_i = 0 \neq b_i, a_i \neq 0 = b_i \text{ or } a_i = b_i = 0\} \tag{36}$$

$$\cup \quad \{\exists l : a_l \neq 0 \neq b_l \text{ and } \forall i \neq l : a_i = b_i = 0\}. \tag{37}$$

This is equivalent to the columns being a multiple of a standard basis vector, one of them is the zero vector or they are orthogonal to each other. To show why this statement holds, we are using induction on k. For $k = 2$, we have the following set of solutions:

$$\underbrace{\begin{pmatrix} 0 & b_1 \\ 0 & b_2 \end{pmatrix} \quad \begin{pmatrix} a_1 & 0 \\ a_2 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 \\ a_2 & b_2 \end{pmatrix} \quad \begin{pmatrix} a_1 & b_1 \\ 0 & 0 \end{pmatrix}}_{\text{Case 1}} \Bigg| \underbrace{\begin{pmatrix} a_1 & 0 \\ 0 & b_2 \end{pmatrix} \quad \begin{pmatrix} 0 & b_1 \\ a_2 & 0 \end{pmatrix}}_{\text{Case 2}}$$

So, the statement holds for $k = 2$.

Next assume we solved the problem for columns with k entries and want to deduce the case where we have $k+1$ entries (i.e. they satisfy the equations in $S(k+1)$). The pair a_{k+1}, b_{k+1} satisfies either one of the three equations in line equation 36: $a_{k+1} = b_{k+1} = 0$, $a_{k+1} = 0 \neq b_{k+1}$ or $a_{k+1} \neq 0 = b_{k+1}$. If $a_{k+1} = b_{k+1} = 0$, the rest of the non-trivial equations will be the same set of equations that one will get in the case of k entries. If $a_{k+1} = 0 \neq b_{k+1}$, we would be left with equation $a_i b_{k+1} = 0$ (Case 1) or $b_i b_{k+1} = 0$ (Case 2) which would mean that for all $i \leq k$ either $a_i = 0$ or $b_i = 0$, which will satisfy the equations in $S(k+1)$. We get an analogous argument in the case of $a_{k+1} \neq 0 = b_{k+1}$.

The other possibility is that $a_{k+1} \neq 0 \neq b_{k+1}$. But in this case both equations from case one and two $a_{k+1} b_i = a_i b_{k+1} = 0$ and $a_{k+1} a_i = b_{k+1} b_i = 0$ lead to $a_i = b_i = 0$ for all $i \leq k$ and this satisfies $S(k+1)$ in line equation 37, concluding the induction.

This completes the proof. Since $\lim_{n\to\infty} s_n \neq 1$ only if $\lim_{n\to\infty} A_n$ has a column that is the zero vector, a multiple of a standard basis vector or it has two columns that are orthogonal to each. Exactly, the conditions excluded in the theorem. For all other cases, we get a contradiction: therefore $\lim_{n\to\infty} s_n = 1$ and M_n converges to a rank-1 matrix. □

B. Convergence Speed & Simulation of Matrix Convergences

We proved that $M_n = \prod_i^n A_i$ converges to a rank-1 matrix for $n \to \infty$, but which practical implications has this for a 16 weight-layered network? How quickly is the convergence for matrices consider in neural networks?

We know that s_n increases by a factor $(1 + c)$ greater than 1 ($c > 0$):

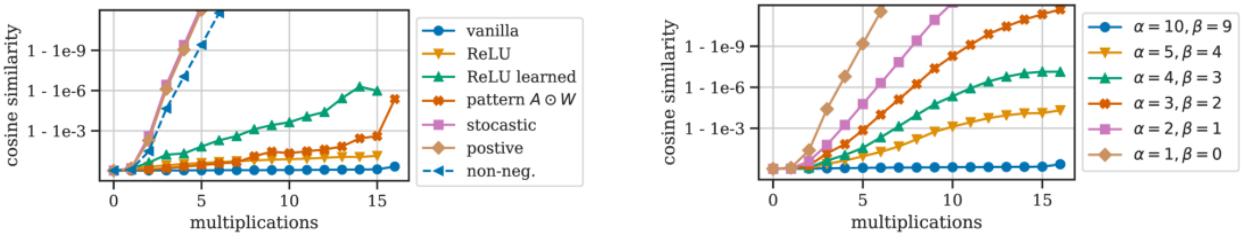
$$s_n \geq (1 + c) s_{n-1} \tag{38}$$

Each iteration yields such a factor and we get a chain of factors:

$$s_n \geq (1 + c_n)(1 + c_{n-1})...(1 + c_2) s_1 \tag{39}$$

Although the multiplication chain of c_n has some similarities to an exponential form: γ^n, s_n does not have to converge exponentially as the individual c_n have to decrease (s_n bounded by 1). We investigated the convergence speed using a simulation of random matrices and find that non-negative matrices decay exponentially fast towards 1.

We report the converging behavior for matrix chains which resembles a VGG-16. As in the backward pass, we start from the last layer. The convolutional kernels are considered to be 1x1, e.g. for a kernel of size (3, 3, 256, 128), we use a matrix of size (256, 128).



(a) different matrix properties (see text)

(b) $\alpha W^+ + \beta W^-$ where $W_{[ij]} \sim \mathcal{N}(0, 1)$

Figure 8: Simulated convergence for a matrix chain.

We test out the effect of different matrix properties. For *vanilla*, we sample the matrix entries from a normal distribution. In the next setting, we apply a *ReLU* operation after each multiplication. For *ReLU learned*, we used the corresponding learned VGG parameters. We generate *non-negative* matrices containing 50% zeros by clipping random matrices to $[0, \infty]$. And *positive* matrices by taking the absolute value. We report the median cosine similarity between the column vectors of the matrix.

The y-axis of figure 8a has logarithmic scale. We observe that the positive, stochastic, and non-negative matrices yield a linear path, indicating an exponential decay of the form: $1 - \exp(-\lambda n)$. The 50% zeros in the non-negative matrices only result in a bit lower convergence slope. After 7 iterations, they converged to a single vector with floating point imprecision.

We also investigated how a slightly negative matrix influence the convergence. In figure 8b, we show the converges of matrices: $\alpha W^+ + \beta W^-$ where $W^+ = \max(0, W), W^- = \min(0, W)$ and $W \sim \mathcal{N}(0, I)$. We find that for small enough $\beta < 4$ values the matrix chains still converge. This simulation motivated us to include $\text{LRP}_{\alpha 5 \beta 4}$ in our evaluation which show less convergence on VGG-16, but its saliency maps also contain more noise.

C. Pattern Attribution

We derive equation 9 from the original equation given in (Kindermans et al., 2018). We will use the notation from the original paper and denote a weight vector with $\boldsymbol{w} = W_{l_{[i,:]}}$ and the corresponding pattern with $\boldsymbol{a} = A_{l_{[i,:]}}$. The output is $y = \boldsymbol{w}^T \boldsymbol{x}$.

Derivation of Pattern Computation For the positive patterns of the two component estimator $S_{\boldsymbol{a}+-}$, the expectation is taken only over $\{\boldsymbol{x} | \boldsymbol{w}^T \boldsymbol{x} > 0\}$. We only show it for the positive pattern $\boldsymbol{a}_+$. As our derivation is independent of the subset of $\boldsymbol{x}$ considered, it would work analogously for negative patterns or the linear estimator $S_{\boldsymbol{a}}$.

The formula to compute the pattern $\boldsymbol{a}_+$ is given by:

$$\begin{aligned} \boldsymbol{a}_+ &= \frac{\mathbb{E}_+ [\boldsymbol{x} y] - \mathbb{E}_+ [\boldsymbol{x}] \mathbb{E}_+ [y]}{\boldsymbol{w}^T \mathbb{E}_+ [\boldsymbol{x} y] - \boldsymbol{w}^T \mathbb{E}_+ [\boldsymbol{x}] \mathbb{E}_+ [y]} \\ &= \frac{\text{cov}[\boldsymbol{x}, \boldsymbol{w}^T \boldsymbol{x}]}{\boldsymbol{w}^T \text{cov}[\boldsymbol{x}, \boldsymbol{w}^T \boldsymbol{x}]}, \end{aligned} \tag{40}$$

where $\text{cov}[\boldsymbol{x}, \boldsymbol{w}^T \boldsymbol{x}] = \mathbb{E}_+[\boldsymbol{x} y] - \mathbb{E}_+[\boldsymbol{x}]\mathbb{E}_+[y]$. Using the bilinearity of the covariance matrix ($\text{cov}[\boldsymbol{b}, \boldsymbol{c}^T \boldsymbol{d}] = \text{cov}[\boldsymbol{b}, \boldsymbol{d}]\boldsymbol{c}$), gives:

$$\boldsymbol{a}_+ = \frac{\text{cov}[\boldsymbol{x}, \boldsymbol{x}] \boldsymbol{w}}{\boldsymbol{w}^T \text{cov}[\boldsymbol{x}, \boldsymbol{x}] \boldsymbol{w}}. \tag{41}$$

Using the notation $\text{cov}[\boldsymbol{h}] = \text{cov}[\boldsymbol{x}, \boldsymbol{x}]$ gives equation 9.

Connection to power iteration A step of the power iteration is given by:

$$\boldsymbol{v}_{k+1} = \frac{M \boldsymbol{v}_k}{\| M \boldsymbol{v}_k \|} \tag{42}$$

The denominator in equation 9 is $\boldsymbol{w}^T \text{cov}[\boldsymbol{h}] \boldsymbol{w}$. Using the symmetry of $\text{cov}[\boldsymbol{h}]$, we have:

$$\left\| \text{cov}[\boldsymbol{h}]^{1/2} \boldsymbol{w} \right\| = (\boldsymbol{w}^T \text{cov}[\boldsymbol{h}]^{1/2} \text{cov}[\boldsymbol{h}]^{1/2} \boldsymbol{w})^{1/2} = (\boldsymbol{w}^T cov[\boldsymbol{h}] \boldsymbol{w})^{1/2} \tag{43}$$

This should be similar to the norm $\|\text{cov}[h]\boldsymbol{w}\|$. As only a single step of the power iteration is performed, the scaling is not too important.

For the denomiator, we therefore have: $\boldsymbol{w}^T cov[\boldsymbol{h}] \boldsymbol{w} = \left\| \text{cov}[\boldsymbol{h}]^{1/2} \boldsymbol{w} \right\|^2$.

This is similar to the norm $\|\text{cov}[h]\boldsymbol{w}\|$. And as only a single step of the power iteration is performed, a different scaling should still yield vectors that are more aligned towards the first eigenvector of $\text{cov}[\boldsymbol{h}]$.

D. CIFAR-10 Network Architecture

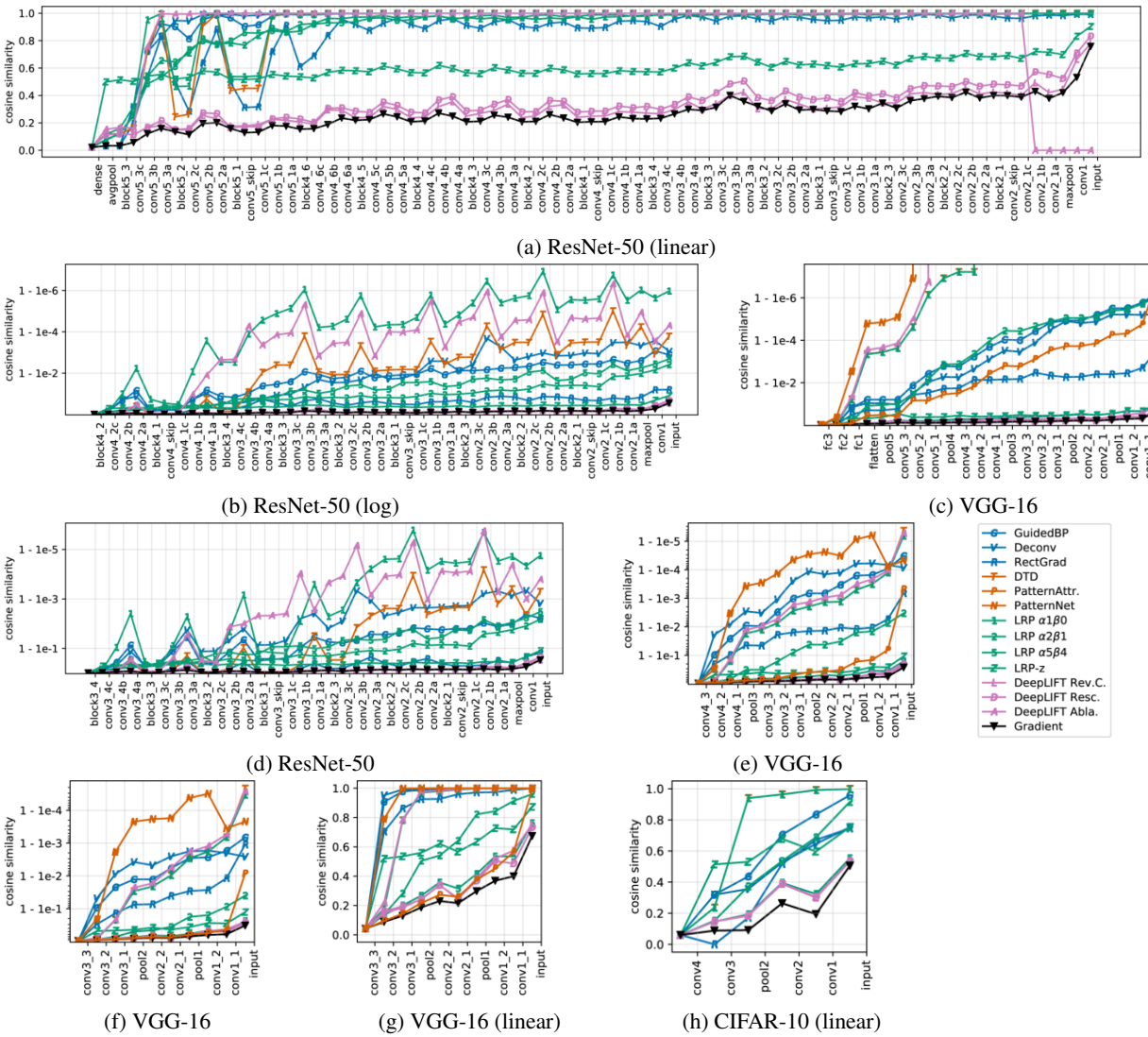
```
# network architecture as a keras model
model = Sequential()

model.add(InputLayer(input_shape=(32, 32, 3), name='input'))
model.add(Conv2D(32, (3, 3), padding='same', name='conv1'))
model.add(Activation('relu', name='relu1'))
model.add(Conv2D(64, (3, 3), padding='same', name='conv2'))
model.add(Activation('relu', name='relu2'))
model.add(MaxPooling2D(pool_size=(2, 2), name='pool2'))

model.add(Conv2D(128, (3, 3), padding='same', name='conv3'))
model.add(Activation('relu', name='relu3'))
model.add(Conv2D(128, (3, 3), padding='same', name='conv4'))
model.add(Activation('relu', name='relu4'))
model.add(MaxPooling2D(pool_size=(2, 2), name='pool4'))

model.add(Flatten(name='flatten'))
model.add(Dropout(0.5, name='dropout5'))
model.add(Dense(1024, name='fc5'))
model.add(Activation('relu', name='relu5'))
model.add(Dropout(0.5, name='dropout6'))
model.add(Dense(10, name='fc6'))
model.add(Activation('softmax', name='softmax'))
```

E. Additional Cosine Similarity Figures



(a) ResNet-50 (linear)

(b) ResNet-50 (log)

(c) VGG-16

(d) ResNet-50

(e) VGG-16

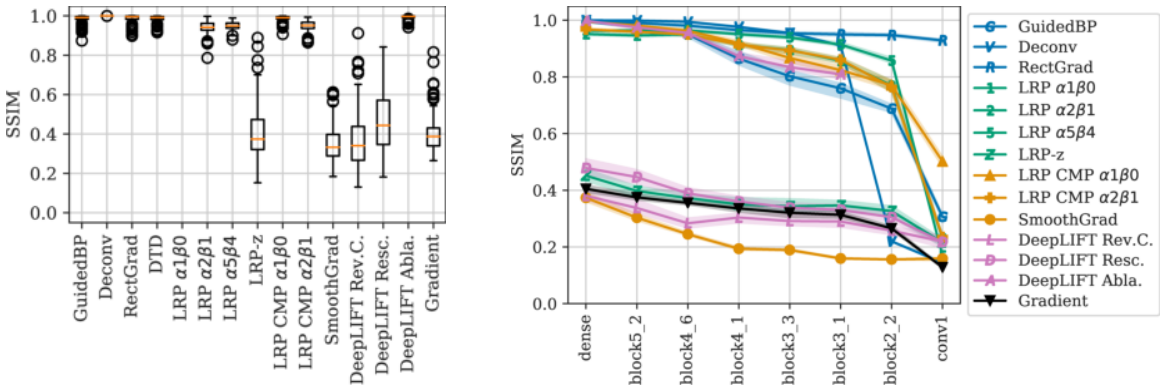
(f) VGG-16

(g) VGG-16 (linear)

(h) CIFAR-10 (linear)

Figure 9: Convergence measured using the CSC for different starting layers.

F. Results on ResNet-50



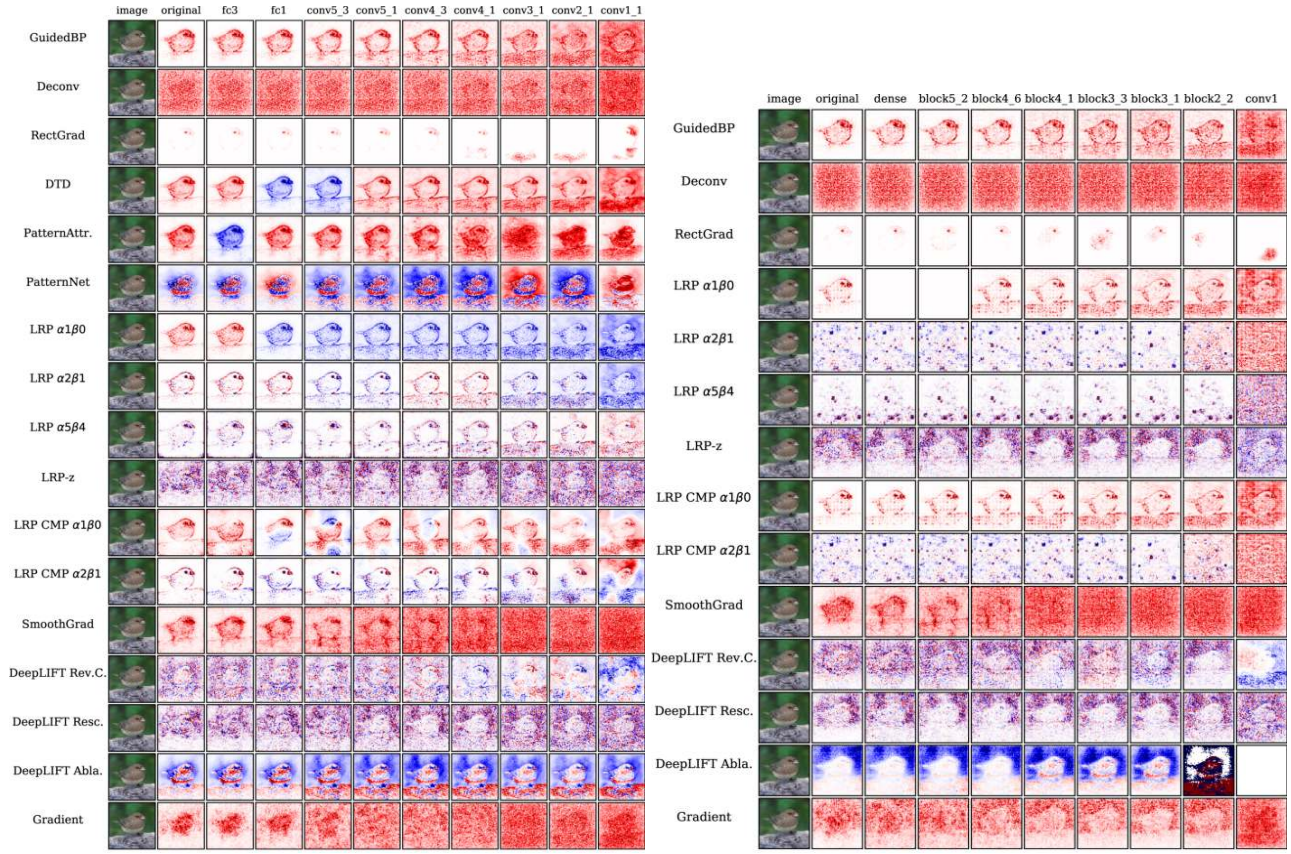
(a) Random Logits

(b) Cascading Parameter Randomization

Figure 10: Effect of (a) randomizing the logits or (b) the parameters on a ResNet-50.

G. Saliency maps for Sanity Checks

For visualization, we normalized the saliency maps to be in $[0, 1]$ if the method produce only positive relevance. If the method also estimates negative relevance, than it is normalized to $[-1, 1]$. The negative and positive values are scaled equally by the absolute maximum. For the sanity checks, we scale all saliency maps to be in $[0, 1]$.



(a) Sanity checks on VGG-16.

(b) Saliency maps on a ResNet-50.