PatternNet and PatternLRP
Improving the interpretability of neural networks

Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Sven Dähne*
Machine Learning Group, TU-Berlin, Germany
p.kindermans@tu-berlin.de

Abstract

Deep learning has significantly advanced the state of the art in machine learning. However, neural networks are often considered black boxes. There is significant effort to develop techniques that explain a classifier's decisions. Although some of these approaches have resulted in compelling visualisations, there is a lack of theory of what is actually explained. Here we present an analysis of these methods and formulate a quality criterion for explanation methods. On this ground, we propose an improved method that may serve as an extension for existing back-projection and decomposition techniques.

1 Introduction

Deep learning [1, 2] has had a huge impact on a wide variety of applications [3, 4, 5, 6]. Current models have millions of parameters spread over many layers with nonlinear activation functions in between. This enables them to learn efficient and powerful representations, but interpretability suffers. Neural networks are therefore considered a 'black-box'. To make them more interpretable many techniques have been proposed [7, 8, 9, 10, 11, 12, 13, 14, 15].

A subset of these methods aim to find input images that maximally activate specific neurons by optimising the image or search through the dataset. [7, 8, 9]. Other methods aim to explain the decision process for a given input image [7, 10, 11, 12, 13, 14, 15]. In this work we examine techniques to explain individual classifier decisions that use back-projections to input space to explain individual decisions of classifiers. This includes model gradients [10] a.k.a. saliency maps [7], DeConvNet (DCN) [13], Guided BackProp (GBP) [14], Layer-wise Relevance Propagation framework (LRP) [11] and the Deep Taylor Decomposition (DTD) [12].

Because linear models are very simple neural networks, we start with an analysis of linear models inspired by Haufe et al. [16] in section 2. This analysis shows that contrary to popular belief, the weights of a linear model are *not* informative about which features are relevant [16]. We discuss existing explanation methods in section 3 and link them to the analysis of linear models. For a linear model DCN and GBP reduce to the gradient, i.e., the weight vector. According to the analysis of linear models, DCN and GBP should not be informative. We show that LRP is sub-optimal because it visualises contributions that are mutually cancelled out. Even though LRP, DCN and GBP do not optimally explain a linear model, they are used to interpret highly complex architectures and produce compelling visualisations (see Fig. 1). Addressing this counter-intuitive issue motivates our work. We propose an objective function to measure the quality of neuron-wise explanations (section 4). Based on this measure we propose PatternNet and PatternLRP. These methods are theoretically sound for linear models and show improved results on deep networks.

Notation and scope Scalars are lowercase letters (i), column vectors are bold ($\boldsymbol{u}$), elementwise multiplication is ($\odot$). The covariance between 2 random vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is $\text{cov}[\boldsymbol{u}, \boldsymbol{v}]$, the covariance

*Klaus-Robert Müller is also with Korea University, Sven Dähne is now at Amazon

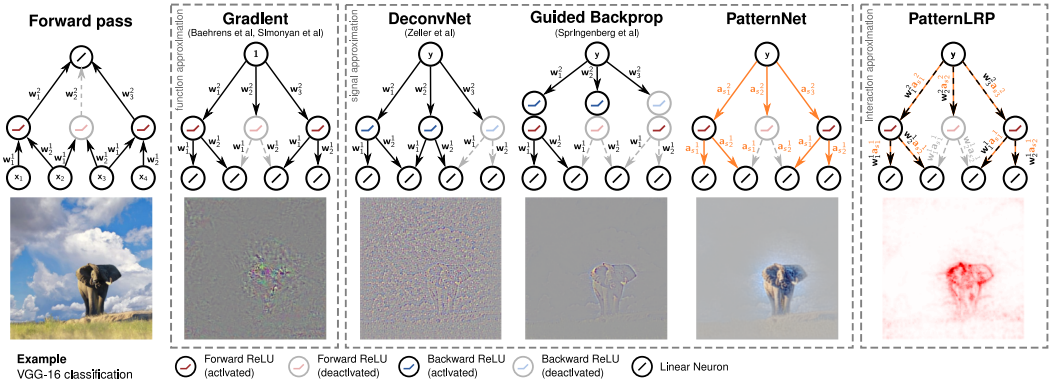

Figure 1: Illustration of explanation approaches. Function and signal approximators visualise the explanation using the original colour channels. The interaction approximator is visualised as a heat map of pixel-wise contributions to the output

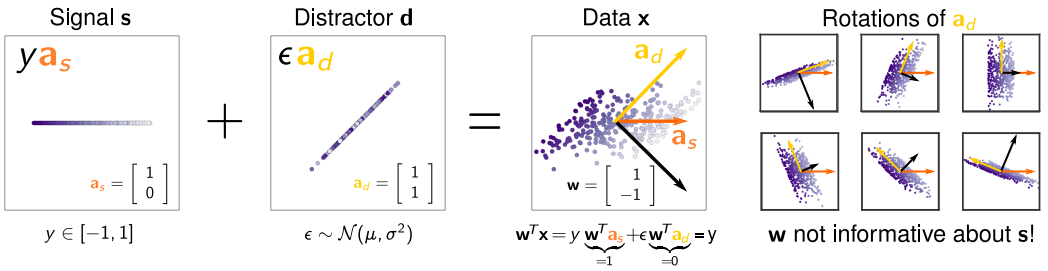

Figure 2: For linear models, i.e., a simple neural network, the weight vector does **not** explain the signal it detects [16]. The data $\boldsymbol{x} = y\boldsymbol{a}_s + \epsilon\boldsymbol{a}_d$ is colour-coded w.r.t. the output $y = \boldsymbol{w}^T\boldsymbol{x}$. Only the signal $s = y\boldsymbol{a}_s$ contributes to y. The weight vector $\boldsymbol{w}$ does not agree with the signal direction, since its primary objective is canceling the distractor. Therefore, rotations of the basis vector $\boldsymbol{a}_d$ of the distractor with constant signal $\boldsymbol{s}$ lead to rotations of the weight vector (**right**).

of $\boldsymbol{u}$ and i is $\mathrm{cov}[\boldsymbol{u}, i]$. The variance of a scalar random variable i is σ_i^2. Estimates of random variables will have a hat ($\hat{\boldsymbol{u}}$). We analyse neural networks excluding the final soft-max output layer. To allow for analytical treatment, we only consider networks with linear neurons optionally followed by a rectified linear unit (ReLU), max-pooling or soft-max. We analyse linear neurons and nonlinearities independently. These restrictions are similar to those in the saliency map [7], DCN [13], GBP [14], LRP [11] and DTD [12]. Without loss of generality, biases are constant neurons.

2 Understanding linear models

Before moving to deep networks, we analyse the behaviour of a simple linear model (see Fig. 2). Consider the following toy example where we generate data $\boldsymbol{x}$ as:

$$\begin{aligned} \boldsymbol{x} = \boldsymbol{s} + \boldsymbol{d} \qquad \boldsymbol{s} &= \boldsymbol{a}_s y, & \text{with } \boldsymbol{a}_s &= (1, 0)^T, & y &\in [-1, 1] \\ \boldsymbol{d} &= \boldsymbol{a}_d \epsilon, & \text{with } \boldsymbol{a}_d &= (1, 1)^T, & \epsilon &\sim \mathcal{N}\left(\mu, \sigma^2\right). \end{aligned}$$

We train a linear regression model to extract y from $\boldsymbol{x}$. By construction, $\boldsymbol{s}$ is the *signal* in our data, i.e., the part of $\boldsymbol{x}$ containing information about y. Using the terminology of [16], the direction $\boldsymbol{a}_s$ along which the signal varies is called the *pattern*. The *distractor* $\boldsymbol{d}$ obfuscates the signal making the detection task more difficult. To optimally extract y, our model has to be able to filter out the distractor $\boldsymbol{d}$. This is why the weight vector is also called the *filter*. In the example, $\boldsymbol{w} = [1, -1]^T$ fulfils this task.

From this simple example, we can make several observations: Most importantly, the optimal weight vector $\boldsymbol{w}$ does *not* align, in general, with the signal direction $\boldsymbol{a}_s$ (see Fig. 2). Moreover, when the direction of the distractor $\boldsymbol{a}_d$ changes, $\boldsymbol{w}$ must follow, as illustrated on the right hand side of the

figure. On the other hand, a change in signal direction $\boldsymbol{a}_s$ can be compensated for by a change in sign and magnitude of $\boldsymbol{w}$ such that $\boldsymbol{w}^T\boldsymbol{a}_s = 1$; but the direction stays constant. The fact that the direction of the weight vector in a linear model is largely determined by the distractor is essential to understand how linear models operate. This property implies that given only the weight vector, we cannot know what part of the data is informative about the output y. This is the direction $\boldsymbol{a}_s$ and must be learned from data.

The following terminology is used throughout this manuscript: The *filter* $\boldsymbol{w}$ tells us how to extract the output y optimally from data $\boldsymbol{x}$. The *pattern* $\boldsymbol{a}_s$ is the direction in the data along which the desired output y varies. Both constitute the *signal* $\boldsymbol{s} = \boldsymbol{a}_s y$, i.e., the informative part of $\boldsymbol{x}$. The *distractor* $\boldsymbol{d}$ is the component of the data that does not contain information about the desired output.

3 Overview of explanation approaches and their behaviour

Given our analysis of linear models, we can take a look at a subset of explanation methods for individual classifier decisions. We introduce general explanation concepts and discuss how they are connected to our analysis of linear models in the previous section. Fig. 1 gives an overview of the different types of explanation methods discussed below. The different approaches can be divided into: function approximation, signal approximation and interaction approximation. These three groups present different information about the network that is analysed. Therefore these groups complement each other.

Functions The function being modelled by a neural network extracts the signal from the data. Explaining the function in input space corresponds to describing how the function extracts y from $\boldsymbol{x}$. Since deep neural networks are highly nonlinear, this can only be approximated. The saliency map estimates how moving along a particular direction in input space influences y (i.e., sensitivity analysis) where the direction is given by the model gradient. In case of a linear model $y = \boldsymbol{w}^T\boldsymbol{x}$, the saliency map reduces to analysing the filters

$$\partial y / \partial \boldsymbol{x} = \boldsymbol{w}.$$

Since it is mostly determined by the distractor as demonstrated above, it is not informative about the signal. It tells us how to extract the signal, not what the signal is.

Signal The signal $\boldsymbol{s}$ detected by the neural network is the informative component of the data w.r.t. the network. Finding it is challenging and the goal of this subset of explanation methods. In a linear model, the signal corresponds to

$$\boldsymbol{s} = \boldsymbol{a}_s y.$$

The pattern $\boldsymbol{a}_s$ contains the information, i.e., it tells us where a change of the output variable is expected to be measurable in the input [16]. Many have attempted to visualise the signal for deep neural networks using DeConvNet [13] and Guided BackProp [14]. These use the same algorithm as the saliency map, but treat the rectifiers differently (see Fig. 1): DeConvNet leaves out the rectifiers from the forward pass, but adds additional ReLUs after each deconvolution, while Guided BackProp uses the ReLUs from the forward pass as well as additional ones. The back-projections for the linear components of the network correspond to a superposition of what are assumed to be the signal directions of each neuron. For this reason, these projections must be seen as an approximation of the features that activated the higher layer neuron. It is not a reconstruction in input space [13]. Unfortunately, for the simplest of neural networks, the linear model, these visualisations reduce to the gradient. They show the filter $\boldsymbol{w}$ and *neither* the pattern $\boldsymbol{a}_s$, *nor* the signal $\boldsymbol{s}$. Hence, DeConvNet and Guided BackProp do not guarantee to produce the detected signal for a linear model, which is proven by our toy example in Fig. 2. Since they do produce compelling visualisations, we will investigate whether the direction of the filter $\boldsymbol{w}$ coincides with the direction of the signal $\boldsymbol{s}$. We show later that this is *not* the case and propose a new approach to learn the correct direction that improves upon the DeConvNet and Guided BackProp visualisations, namely the PatternNet in Fig. 1.

Function-signal interaction Finally, we can look at how the signal dimensions contribute to the output through the layers. This will be referred to as the *interaction* visualisation or heat map. For a linear model, the optimal heat map would be obtained by element-wise multiplying the signal with the weight vector:

$$\boldsymbol{r}^{input} = \boldsymbol{w} \odot \boldsymbol{a} y,$$

with $\odot$ the elementwise multiplication. Bach et al. [11] introduced *layer-wise relevance propagation* (LRP) as a decomposition of pixel-wise contributions (called *relevances*). Montatavon et al. [12] extended this idea and proposed the deep Taylor decomposition (DTD). The key idea of DTD is to decompose the activation of a neuron in terms of contributions from its inputs. This is achieved using a first-order Taylor expansion around a root point $\boldsymbol{x}_0$ with $\boldsymbol{w}^T\boldsymbol{x}_0 = 0$. The relevance of the selected output neuron i is initialised with its output from the forward pass:

$$r_i^{output} = y.$$

Then the relevance is re-distributed towards the input, where a non-active ReLu unit from the forward pass stops the re-distribution. This is identical to how a ReLu stops the propagation of the gradient. For a linear layer, the relevance r_i^l of neuron i in layer l is re-distributed using:

$$\boldsymbol{r}^{l-1,i} = \frac{\boldsymbol{w} \odot (\boldsymbol{x} - \boldsymbol{x}_0)}{\boldsymbol{w}^T\boldsymbol{x}} r_i^l.$$

Here we can safely assume that $\boldsymbol{w}^T\boldsymbol{x} > 0$, since its subsequent ReLu must have been activated in the forward pass if it is used in the backward pass. The difficulty in application of the deep Taylor decomposition is the choice of the root point $\boldsymbol{x}_0$, for which many options are available. It is important to recognise at this point that selecting a root point for the DTD corresponds to estimating the distractor $\boldsymbol{x}_0 = \boldsymbol{d}$ and, by that, the signal

$$\hat{\boldsymbol{s}} = \boldsymbol{x} - \boldsymbol{x}_0.$$

Summarising, the **function** extracts the **signal** from the data by removing the distractor. The **interaction** of function and signal determines how much an individual component of the signal contributes to the output, which is what LRP calls *relevance*.

4 PatternNet and PatternLRP: learning to estimate the signal

Visualising the function has proven to be straightforward [10, 7]. In contrast, visualising the signal [16, 13, 14] and the function-signal interaction [11, 12] is more difficult. It requires a good estimate of what is the signal and what is the distractor. In the following section we first propose a quality measure for neuron-wise signal estimators. This allows us to evaluate existing approaches and, finally, derive signal estimators that optimise this criterion. These estimators will then be used to explain the signal (PatternNet) and the signal-function interaction (PatternLRP). All mentioned explanation techniques as well as our proposed signal estimators treat neurons independently, i.e., the full explanation will be a superposition of neuron-wise explanations.

4.1 Quality criterion for signal estimators

Recall that the input data $\boldsymbol{x}$ comprises both signal and distractor: $\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{d}$, and that the signal contributes to the output but the distractor does not. Assuming the filter $\boldsymbol{w}$ has been trained sufficiently well to extract y, we have

$$\boldsymbol{w}^T\boldsymbol{x} = y, \quad \boldsymbol{w}^T\boldsymbol{s} = y, \quad \boldsymbol{w}^T\boldsymbol{d} = 0.$$

Note that estimating the signal based on these conditions alone is an ill-posed problem. Even if we limit ourselves to linear estimators of the form $\hat{\boldsymbol{s}} = \boldsymbol{u}(\boldsymbol{w}^T\boldsymbol{u})^{-1}y$, an infinite number of options remain: Let $\boldsymbol{u}$ be any vector such that $\boldsymbol{w}^T\boldsymbol{u} \neq 0$ then each signal estimate $\hat{\boldsymbol{s}} = \boldsymbol{u}\left(\boldsymbol{w}^T\boldsymbol{u}\right)^{-1}y$ satisfies $\boldsymbol{w}^T\hat{\boldsymbol{s}} = y$. This implies the existence of an infinite number of possible rules for the DTD as well as infinitely many back-projections for the DeConvNet family.

To alleviate this issue, we introduce the following quality measure ρ for a signal estimator $S(\boldsymbol{x}) = \hat{\boldsymbol{s}}$ that will be written with explicit variances and covariances using the shorthands $\hat{\boldsymbol{d}} = \boldsymbol{x} - S(\boldsymbol{x})$ and $y = \boldsymbol{w}^T\boldsymbol{x}$:

$$\rho(S) = 1 - \max_{\boldsymbol{v}} corr\left(\boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{v}^T\left(\boldsymbol{x} - S(\boldsymbol{x})\right)\right) = 1 - \max_{\boldsymbol{v}} \frac{\boldsymbol{v}^T \mathrm{cov}[\hat{\boldsymbol{d}}, y]}{\sqrt{\sigma^2_{\boldsymbol{v}^T\hat{\boldsymbol{d}}}\sigma^2_y}}. \tag{1}$$

This criterion introduces an additional constraint by measuring how much information about y can be reconstructed from the residuals $\boldsymbol{x} - \hat{\boldsymbol{s}}$ using a linear projection. The best signal estimators

remove most of the information in the residuals and thus yield large $\rho(S)$. Since the correlation is invariant to scaling, we constrain $\boldsymbol{v}^T\boldsymbol{d}$ to have variance $\sigma^2_{\boldsymbol{v}^T\hat{\boldsymbol{d}}} = \sigma^2_{\boldsymbol{w}^T\boldsymbol{x}}$. Finding the optimal $\boldsymbol{v}$ for a fixed $S(\boldsymbol{x})$ amounts to a least-squares regression from $\hat{\boldsymbol{d}}$ to y. This enables us to assess the quality of signal estimators efficiently.

4.2 Existing Signal Estimators

Let us now discuss two signal estimators that have been used in previous approaches.

$S_{\boldsymbol{x}}$ – the identity estimator The naive approach to signal estimation is to assume the entire data is signal and there are no distractors:

$$S_{\boldsymbol{x}}(\boldsymbol{x}) = \boldsymbol{x}.$$

With this being plugged into the deep Taylor framework, we obtain the z-rule [12] which corresponds to standard LRP [11]. For a linear model, this corresponds to $\boldsymbol{w} \odot \boldsymbol{x}$ as a function-signal interaction explanation. It can be shown that for ReLu and max-pooling networks, the z-rule reduces to the element-wise multiplication of the input and the saliency map. This means that for a whole network, the assumed signal is simply the original input image. It also implies that, if there are distractors present in the data, they are included in the function-signal interaction:

$$\boldsymbol{r} = \boldsymbol{w} \odot \boldsymbol{x} = \boldsymbol{w} \odot \boldsymbol{s} + \boldsymbol{w} \odot \boldsymbol{d}.$$

When moving through the layers by applying the filters $\boldsymbol{w}$ during the forward pass, the contributions from the distractor $\boldsymbol{d}$ are cancelled out. Ideally the explanation should be limited to contributions from the signal $\boldsymbol{s}$:

$$\boldsymbol{r} = \boldsymbol{w} \odot \boldsymbol{s}.$$

However, they cannot be cancelled in the backward pass by the element-wise multiplication $\boldsymbol{w} \odot \boldsymbol{x}$, explaining the noisy nature of the visualisations based on the z-rule.

$S_{\boldsymbol{w}}$ – the filter based estimator The implicit assumption made by DeConvNet and Guided BackProp is that the detected signal varies in the direction of the weight vector $\boldsymbol{w}$. This weight vector has to be normalised in order to be a valid signal estimator. In the deep Taylor decomposition framework this corresponds to the $\boldsymbol{w}^2$-rule and results in the following signal estimator:

$$S_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}}\boldsymbol{w}^T\boldsymbol{x}.$$

For a linear model, this produces function-signal interaction of the form $\frac{\boldsymbol{w}\odot\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}}y$. This estimator fails to reconstruct the proper signal in the toy example of Section 2.

4.3 PatternNet and PatternLRP

We suggest to learn the signal estimator S from data by optimising the previously established criterion. A signal estimator S is optimal with respect to Eq. (1) if the correlation is zero for all possible $\boldsymbol{v}$: $\forall \boldsymbol{v}, \text{cov}[y, \hat{\boldsymbol{d}}]\boldsymbol{v} = \boldsymbol{0}$. This is the case when there is no covariance between y and $\hat{\boldsymbol{d}}$. Because of linearity of the covariance and since $\hat{\boldsymbol{d}} = \boldsymbol{x} - S(\boldsymbol{x})$ the above condition leads to

$$\text{cov}[y, \hat{\boldsymbol{d}}] = \boldsymbol{0} \Rightarrow \text{cov}[\boldsymbol{x}, y] = \text{cov}[S(\boldsymbol{x}), y]. \tag{2}$$

It is important to recognise that the covariance is a summarising statistic and consequently the problem can still be solved in multiple ways. We will present two possible solutions to this problem. Note that when optimising the estimator, the contribution from the bias neuron will be considered 0 since it does not covary with the output y.

$S_{\boldsymbol{a}}$ – The linear estimator A linear neuron can only extract linear signals $\boldsymbol{s}$ from its input $\boldsymbol{x}$. Therefore, we could assume a linear dependency between $\boldsymbol{s}$ and y, yielding a signal estimator:

$$S_{\boldsymbol{a}}(\boldsymbol{x}) = \boldsymbol{a}\boldsymbol{w}^T\boldsymbol{x}. \tag{3}$$

Plugging this into Eq. (2) and optimising for $\boldsymbol{a}$ yields

$$\text{cov}[\boldsymbol{x}, y] = \text{cov}[\boldsymbol{a}\boldsymbol{w}^T\boldsymbol{x}, y] = \boldsymbol{a}\text{cov}[y, y] \Rightarrow \boldsymbol{a} = \frac{\text{cov}[\boldsymbol{x}, y]}{\sigma_y^2}.$$

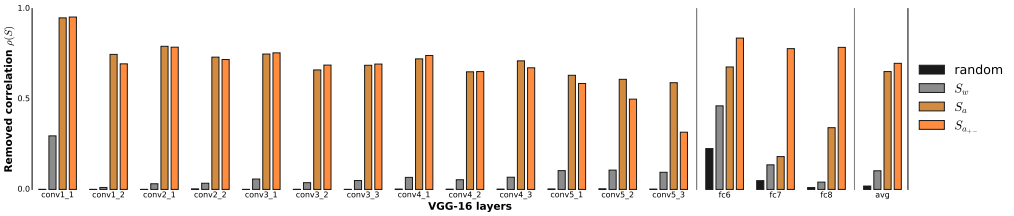

Figure 3: Evaluating $\rho(S)$ for VGG-16 on ImageNet. Higher values are better. The gradient ($S_{\boldsymbol{w}}$), linear estimator ($S_{\boldsymbol{a}}$) and nonlinear estimator ($S_{\boldsymbol{a}_{+-}}$) are compared. An estimator using random directions is the baseline. The network has 5 blocks with 2/3 convolutional layers and 1 max-pooling layer each, followed by 3 dense layers.

Note that this solution is equivalent to the approach commonly used in neuro-imaging [16] despite different derivation. With this approach we can recover the signal of our toy example in Section 2. It is equivalent to the filter-based approach only if the distractors are orthogonal to the signal.

We found that the linear estimator works well for the convolutional layers. However, when using this signal estimator with ReLUs in the dense layers, there is still a considerable correlation left in the distractor component (see Fig. 3).

$S_{\boldsymbol{a}_{+-}}$ – The two-component estimator To move beyond the linear signal estimator, it is crucial to understand how the rectifier influences the training. Since the gate of the ReLU closes for negative activations, the weights only need to filter the distractor component of neurons with $y > 0$. Since this allows the neural network to apply filters locally, we cannot assume a global distractor component. We rather need to distinguish between the positive and negative regime:

$$\boldsymbol{x} = \begin{cases} \boldsymbol{s}_+ + \boldsymbol{d}_+ & \text{if } y > 0 \\ \boldsymbol{s}_- + \boldsymbol{d}_- & \text{otherwise} \end{cases}$$

Even though signal and distractor of the negative regime are canceled by the following ReLU, we still need to make this distinction in order to approximate the signal. Otherwise, information about whether a neuron fired would be retained in the distractor. Thus, we propose the two-component signal estimator:

$$S_{\boldsymbol{a}+-}(\boldsymbol{x}) = \begin{cases} \boldsymbol{a}_+ \boldsymbol{w}^T \boldsymbol{x}, & \text{if } \boldsymbol{w}^T \boldsymbol{x} > 0 \\ \boldsymbol{a}_- \boldsymbol{w}^T \boldsymbol{x}, & \text{otherwise} \end{cases} \tag{4}$$

Next, we derive expressions for the patterns $\boldsymbol{a}_+$ and $\boldsymbol{a}_-$. We denote expectations over $\boldsymbol{x}$ within the positive and negative regime with $\mathbb{E}_+[\boldsymbol{x}]$ and $\mathbb{E}_-[\boldsymbol{x}]$, respectively. Let π_+ be the expected ratio of inputs $\boldsymbol{x}$ with $\boldsymbol{w}^T \boldsymbol{x} > 0$. The covariance of data/signal and output become:

$$\text{cov}[\boldsymbol{x}, y] = \quad \pi_+ \left(\mathbb{E}_+[\boldsymbol{x}y] - \mathbb{E}_+[\boldsymbol{x}]\,\mathbb{E}[y]\right) \; + \quad (1-\pi_+)\left(\mathbb{E}_-[\boldsymbol{x}y] - \mathbb{E}_-[\boldsymbol{x}]\,\mathbb{E}[y]\right) \tag{5}$$

$$\text{cov}[\boldsymbol{s}, y] = \quad \pi_+ \left(\mathbb{E}_+[\boldsymbol{s}y] - \mathbb{E}_+[\boldsymbol{s}]\,\mathbb{E}[y]\right) \; + \quad (1-\pi_+)\left(\mathbb{E}_-[\boldsymbol{s}y] - \mathbb{E}_-[\boldsymbol{s}]\,\mathbb{E}[y]\right) \tag{6}$$

Assuming both covariances are equal, we can treat the positive and negative regime separately using Eq. (2) to optimise the signal estimator:

$$\mathbb{E}_+[\boldsymbol{x}y] - \mathbb{E}_+[\boldsymbol{x}]\,\mathbb{E}[y] \quad = \quad \mathbb{E}_+[\boldsymbol{s}y] - \mathbb{E}_+[\boldsymbol{s}]\,\mathbb{E}[y]$$

Plugging in Eq. (4) and solving for $\boldsymbol{a}_+$ yields the required parameter ($\boldsymbol{a}_-$ analogous).

$$\boldsymbol{a}_+ \quad = \quad \frac{\mathbb{E}_+[\boldsymbol{x}y] - \mathbb{E}_+[\boldsymbol{x}]\,\mathbb{E}[y]}{\boldsymbol{w}^T\,\mathbb{E}_+[\boldsymbol{x}y] - \boldsymbol{w}^T\,\mathbb{E}_+[\boldsymbol{x}]\,\mathbb{E}[y]} \tag{7}$$

PatternNet and PatternLRP Based on the presented analysis, we propose PatternNet and PatternLRP as illustrated in Fig. 1. *PatternNet* yields a layer-wise back-projection of the estimated signal to input space. In each layer, the signal estimator is approximated as a superposition of neuron-wise, nonlinear signal estimators $S_{\boldsymbol{a}+-}$. This can be considered an improvement over DeConvNet and Guided Backprop. *PatternLRP* exposes the function-signal interaction $\boldsymbol{w} \odot \boldsymbol{a}_+$ and improves upon the layer-wise relevance propagation (LRP) framework [11]. It can be seen as a root point estimator for the deep Taylor decomposition (DTD). Here, the explanation consists of neuron-wise contributions of the estimated *signal* to the classification score. By ignoring the distractor, PatternLRP can reduce the noise and produces much clearer heat maps.

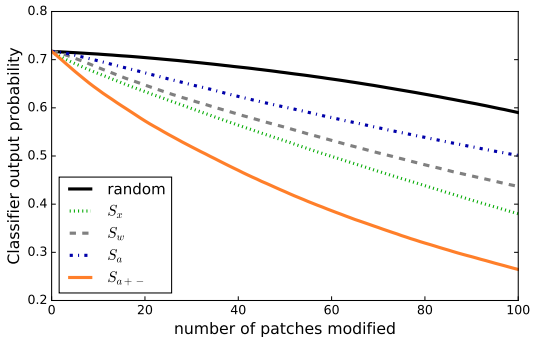

Figure 4: Image degradation experiment on all 50.000 images in the ImageNet validation set. The effect on the classifier output is measured. A steeper decrease is better

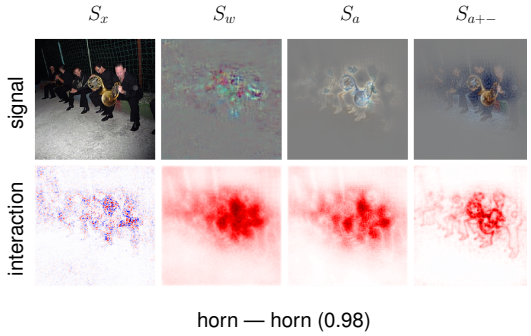

Figure 5: The top row: signal based visualisation, the bottom row: interaction visualisation. Please note that for the trivial estimator S_x the input image is the signal, which does not present more information.

5 Experiments

To demonstrate the quality of the explanations generated by our model, we focused on the task of image classification. Nevertheless, our method is not restricted to networks operating on image inputs. All experiments are implemented using Theano [17] and Lasagne [18]. In this section we restrict the analysis to the well-known ImageNet dataset [19] and the pre-trained VGG-16 model [20]. Images were rescaled and cropped to 224 by 224 pixels.

The signal estimators are trained on the first half of the training dataset. The vector $\boldsymbol{v}$, used to measure the quality of the signal estimator $\rho(\boldsymbol{x})$ in Eq. (1), is optimised on the second half of the training dataset. This enables us to test the signal estimators for generalisation. All the results presented here were obtained using the official validation set of 50000 samples. The validation set was not used for training the signal estimators, nor for training the vector $\boldsymbol{v}$ to measure the quality. Consequently our results are obtained on previously unseen data.

The linear and the two component signal estimators are obtained by solving their respective closed form solutions (Eq. (4) and Eq. (7)). With a highly parallelised implementation using 4 GPUs this could be done in 3-4 hours. This can be considered reasonable given that several days are required to train the actual network. Solving Eq. (1) with the closed form solution is computationally prohibitive since it must be repeated for every single weight vector in the network. Therefore we optimise the equivalent least-squares problem using stochastic mini-batch gradient descent with ADAM [21] until convergence. This was implemented on a single NVIDIA Tesla K40 and took about 24 hours for an entire network.

Measuring the quality of signal estimators In Fig. 3 we present the results from the correlation measure $\rho(\boldsymbol{x})$, higher values are better. We use random directions as baseline signal estimators. Clearly, this approach removes almost no correlation. The filter-based estimator $S_{\boldsymbol{w}}$ succeeds in removing some of the information in the first layer. This indicates that the filters are similar to the

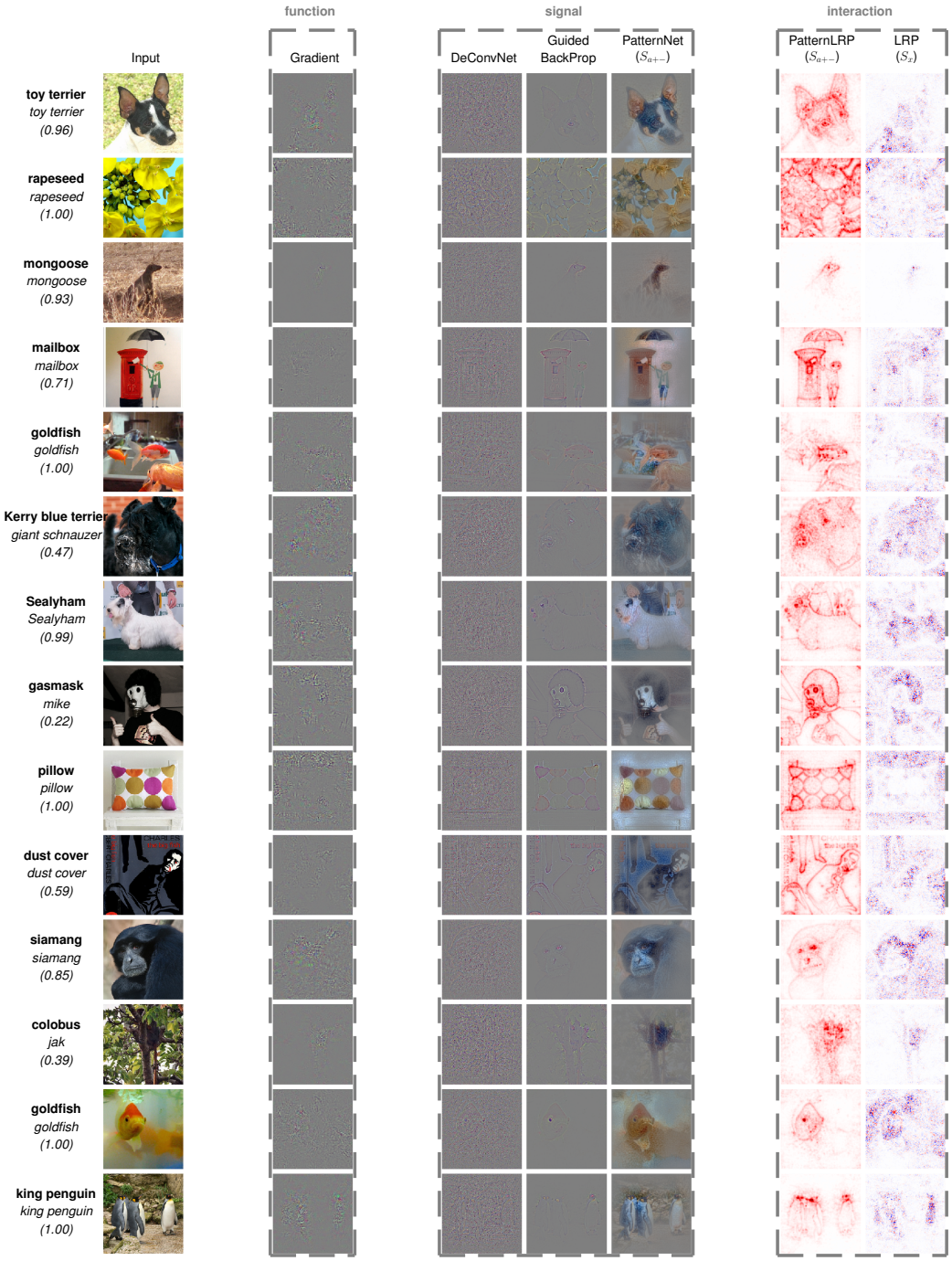

Figure 6: Visualisation of randomly selected images from the ImageNet validation set. Ground truth is shown on top, classifier prediction beneath it. Explanation techniques are grouped by: function, signal or interaction based explanation. The different groups are complementary and methods should only be compared within their group. PatternNet, Guided Backprop, DeConvNet and the Gradient (saliency map) are back-projections to input space with the original colour channels. They are normalised using $x_{norm} = \frac{x}{2\,\mathrm{max}|x|} + \frac{1}{2}$ to maximise contrast. LRP and PatternLRP are heat maps showing pixel-wise contributions to the output. Best viewed in electronic format (zoomed in).

patterns in this layer. However, the gradient removes much less information in the higher layers. Overall, it does not perform much better than the random estimator. This indicates that it cannot provide much insight on how neural networks operate. The optimised estimators remove much more of the correlations across the board. For the convolutional layers, $S_{\boldsymbol{a}}$ and $S_{\boldsymbol{a}+-}$ perform comparably in all but one layer. The two component estimator $S_{\boldsymbol{a}+-}$ outperforms the more basic signal estimator in the dense layers. It also performs best in our second quantitative experiment.

Image degradation Our first experiment was a direct measurement of the quality of the signal estimators of individual neurons. Our second experiment is an indirect measurement of the quality, but it does consider the whole network. We measure how the prediction of the network (after the soft-max) for the initially selected class changes as a function of corrupting more and more patches based on the ordering assigned by the interaction heat map (see [22]). In the experiment, we divide the image in non-overlapping patches of 9x9 pixels. We compute the interaction heat map and sum all the values within a patch. Then we sort the patches in decreasing order based on the aggregate heat map value. In step $n = 1..100$ we replace the first n patches with the their mean per colour channel to remove the information in this patch. Then, we measure how this influences the classifiers output. We use the estimators from the previous experiment to obtain the function-signal interaction heat maps for evaluation. A steeper decay indicates a better heat map. Results are shown in Fig. 4.

Our baseline, in which the patches are randomly ordered performs worst. All signal estimators perform better, however, the linear optimised estimator $S_{\boldsymbol{a}}$ performs quite poorly, followed by the filter-based estimator $S_{\boldsymbol{w}}$. The trivial signal estimator $S_{\boldsymbol{x}}$ performs just slightly better. However, the two component model $S_{\boldsymbol{a}+-}$ leads to the fastest decrease in confidence in the original prediction by a large margin. Its excellent quantitative performance is backed up by the visualisations.

Qualitative evaluation In Fig. 5, we compare all signal estimators on a single input image. For the trivial estimator $S_{\boldsymbol{x}}$, the signal is by definition the original input image and, thus, includes the distractor. Therefore, its noisy interaction heat map shows contributions that cancel each other in the neural network. The $S_{\boldsymbol{w}}$ estimator captures some of the structure. The optimised estimator $S_{\boldsymbol{a}}$ captures slightly more structure but struggles on colour information and produces dense heat maps. The two component model $S_{\boldsymbol{a}+-}$ on the right captures the original input well in the signal estimation and produces a crisp heat map in the interaction estimation.

Fig. 6 shows the visualisations for 14 randomly selected images. PatternNet is able to recover a signal close to the original without having to resort to the inclusion of additional rectifiers in contrast to DeConvNet and Guided BackProp. We argue that this is due to the fact that the optimisation of the pattern allows for capturing the important directions in input space. This contrasts with the commonly used methods DeConvNet, Guided BackProp, LRP and DTD, for which the correlation experiment indicates that their implicit signal estimator cannot capture the true signal in the data. Overall, the proposed, novel approach produces the most crisp visualisation in addition to being measurably better, as shown in the previous section.

6 Conclusion

Understanding and explaining nonlinear methods is an important challenge in machine learning. Algorithms for visualising nonlinear models have emerged but theoretical contributions are scarce. We showed that the direction of a linear neural network's weight vector does not necessarily provide an estimate for the informative (i.e. signal) part of the data. Instead it reflects the relation between the informative direction and the distracting noise contributions (Fig. 2). This implies that popular explanation approaches for neural networks (DeConvNet, Guided BackProp, LRP), do not provide the optimal explanation for a simple linear model. Our reasoning can be extended to nonlinear models. Therefore we propose an objective function for neuron-wise explanations. This can be optimised to correct the signal visualisations (PatternNet) and the decomposition methods (PatternLRP). We demonstrate that our methods are a theoretical, qualitative and quantifiyable improvement over previous methods.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement NO 657679, the BMBF for

the Berlin Big Data Center BBDC (01IS14013A), a hardware donation from NVIDIA. We thank Sander Dieleman, Jonas Degraeve, Ira Korshunova for their comments to improve this manuscript and Gregoire Montavon, Lionel Pigou and the IDLAB members for the valuable discussions and feedback.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[7] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

[8] Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *ICML Workshop on Deep Learning*, 2015.

[9] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[10] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[11] 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*, 10(7):e0130140, 2015.

[12] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[13] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[14] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR*, 2015.

[15] Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR*, 2017.

[16] Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *Neuroimage*, 87:96–110, 2014.

[17] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

[18] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, et al. Lasagne: First release. *Zenodo: Geneva, Switzerland*, 2015.

[19] 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. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[22] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.