# Methods for Interpreting and Understanding Deep Neural Networks

Grégoire Montavon[a,*], Wojciech Samek[b,*], Klaus-Robert Müller[a,c,d,*]

[a] *Department of Electrical Engineering & Computer Science, Technische Universität Berlin, Marchstr. 23, Berlin 10587, Germany*
[b] *Department of Video Coding & Analytics, Fraunhofer Heinrich Hertz Institute, Einsteinufer 37, Berlin 10587, Germany*
[c] *Department of Brain & Cognitive Engineering, Korea University, Anam-dong 5ga, Seongbuk-gu, Seoul 136-713, South Korea*
[d] *Max Planck Institute for Informatics, Stuhlsatzenhausweg, Saarbrücken 66123, Germany*



**Abstract**

This paper provides an entry point to the problem of interpreting a deep neural network model and explaining its predictions. It is based on a tutorial given at ICASSP 2017. It introduces some recently proposed techniques of interpretation, along with theory, tricks and recommendations, to make most efficient use of these techniques on real data. It also discusses a number of practical applications.

*Keywords:* deep neural networks, activation maximization, sensitivity analysis, Taylor decomposition, layer-wise relevance propagation

## 1. Introduction

Machine learning techniques such as deep neural networks have become an indispensable tool for a wide range of applications such as image classification, speech recognition, or natural language processing. These techniques have achieved extremely high predictive accuracy, in many cases, on par with human performance.

In practice, it is also essential to verify for a given task, that the high measured accuracy results from the use of a proper problem representation, and not from the exploitation of artifacts in the data [29, 46, 27]. Techniques for interpreting and understanding what the model has learned have therefore become a key ingredient of a robust validation procedure [51, 6, 5]. Interpretability is especially important in applications such as medicine or self-driving cars, where the reliance of the model on the correct features must be guaranteed [15, 14].

It has been a common belief, that simple models provide higher interpretability than complex ones. Linear models or basic decision trees still dominate in many applications for this reason. This belief is however challenged by recent work, in which carefully designed interpretation techniques have shed light on some of the most complex and deepest machine learning models [44, 55, 5, 37, 40].

Techniques of interpretation are also becoming increasingly popular as a tool for exploration and analysis in the sciences. In combination with deep nonlinear machine learning models, they have been able to extract new insights from complex physical, chemical, or biological systems [20, 21, 49, 43, 54].

This tutorial gives an overview of techniques for interpreting complex machine learning models, with a focus on deep neural networks (DNN). It starts by discussing the problem of interpreting modeled concepts (e.g. predicted classes), and then moves to the problem of explaining individual decisions made by the model. The tutorial abstracts from the exact neural network structure and domain of application, in order to focus on the more conceptual aspects that underlie the success of these techniques in practical applications.

## 2. Preliminaries

Techniques of interpretation have been applied to a wide range of practical problems, and various meanings have been attached to terms such as "understanding", "interpreting", or "explaining". See [32] for a discussion. As a first step, it can be useful to clarify the meaning we associate to these words in this tutorial, as well as the type of techniques that are covered.

We will focus in this tutorial on *post-hoc interpretability*, i.e. a trained model is given and our goal is to understand what the model predicts (e.g. categories) in terms what is readily interpretable (e.g. the input variables) [5, 40]. Post-hoc interpretability should be contrasted to incorporating interpretability directly into the structure of the model, as done, for example, in [39, 15].

Also, when using the word "understanding", we refer to a *functional understanding* of the model, in contrast to a lower-level mechanistic or algorithmic understanding of it. That is, we seek to characterize the model's black-box behavior, without however trying to elucidate its inner workings or shed light on its internal representations.

*Corresponding authors
*Email addresses:* gregoire.montavon@tu-berlin.de (Grégoire Montavon), wojciech.samek@hhi.fraunhofer.de (Wojciech Samek), klaus-robert.mueller@tu-berlin.de (Klaus-Robert Müller)

Throughout this tutorial, we will also make a distinction between *interpretation* and *explanation*, by defining these words as follows.

**Definition 1.** *An interpretation is the mapping of an abstract concept (e.g. a predicted class) into a domain that the human can make sense of.*

Examples of domains that are interpretable are images (arrays of pixels), or texts (sequences of words). A human can look at them and read them respectively. Examples of domains that are *not* interpretable are abstract vector spaces (e.g. word embeddings [33]), or domains composed of undocumented input features (e.g. sequences with unknown words or symbols).

**Definition 2.** *An explanation is the collection of features of the interpretable domain, that have contributed for a given example to produce a decision (e.g. classification or regression).*

An explanation can be, for example, a heatmap highlighting which pixels of the input image most strongly support the classification decision [44, 26, 5]. The explanation can be coarse-grained to highlight e.g. which regions of the image support the decision. It can also be computed at a finer grain, e.g. to include pixels and their color components in the explanation. In natural language processing, explanations can take the form of highlighted text [31, 3].

## 3. Interpreting a DNN Model

This section focuses on the problem of interpreting a concept learned by a deep neural network (DNN). A DNN is a collection of neurons organized in a sequence of multiple layers, where neurons receive as input the neuron activations from the previous layer, and perform a simple computation (e.g. a weighted sum of the input followed by a nonlinear activation). The neurons of the network jointly implement a complex nonlinear mapping from the input to the output. This mapping is learned from the data by adapting the weights of each neuron using a technique called error backpropagation [41].

The learned concept that must be interpreted is usually represented by a neuron in the top layer. Top-layer neurons are abstract (i.e. we cannot look at them), on the other hand, the input domain of the DNN (e.g. image or text) is usually interpretable. We describe below how to build a *prototype* in the input domain that is interpretable and representative of the abstract learned concept. Building the prototype can be formulated within the activation maximization framework.

### *3.1. Activation Maximization (AM)*

Activation maximization is an analysis framework that searches for an input pattern that produces a maximum model response for a quantity of interest [9, 17, 44].

Consider a DNN classifier mapping data points $\boldsymbol{x}$ to a set of classes $(\omega_c)_c$. The output neurons encode the modeled class probabilities $p(\omega_c|\boldsymbol{x})$. A prototype $\boldsymbol{x}^\star$ representative of the class $\omega_c$ can be found by optimizing:

$$\max_{\boldsymbol{x}} \ \log p(\omega_c|\boldsymbol{x}) - \lambda\|\boldsymbol{x}\|^2.$$

The class probabilities modeled by the DNN are functions with a gradient [11]. This allows for optimizing the objective by gradient ascent. The rightmost term of the objective is an $\ell_2$-norm regularizer that implements a preference for inputs that are close to the origin. When applied to image classification, prototypes thus take the form of mostly gray images, with only a few edge and color patterns at strategic locations [44]. These prototypes, although producing strong class response, look in many cases unnatural.

### *3.2. Improving AM with an Expert*

In order to focus on more probable regions of the input space, the $\ell_2$-norm regularizer can be replaced by a data density model $p(\boldsymbol{x})$ called "expert", leading to the new optimization problem:

$$\max_{\boldsymbol{x}} \ \log p(\omega_c|\boldsymbol{x}) + \log p(\boldsymbol{x}).$$

Here, the prototype is encouraged to simultaneously produce strong class response and to resemble the data. By application of the Bayes' rule, the newly defined objective can be identified, up to modeling errors and a constant term, as the class-conditioned data density $p(\boldsymbol{x}|\omega_c)$. The learned prototype thus corresponds to the most likely input $\boldsymbol{x}$ for class $\omega_c$. A possible choice for the expert is the Gaussian RBM [23]. Its probability function can be written as:

$$\log p(\boldsymbol{x}) = \textstyle\sum_j f_j(\boldsymbol{x}) - \frac{1}{2}\boldsymbol{x}^\top\Sigma^{-1}\boldsymbol{x} + \text{cst}.$$

where $f_j(\boldsymbol{x}) = \log(1 + \exp(\boldsymbol{w}_j^\top\boldsymbol{x} + b_j))$ are factors with parameters learned from the data. When interpreting concepts such as natural images classes, more complex density models such as convolutional RBM/DBMs [28], or pixelRNNs [52] are needed.

In practice, the choice of the expert $p(\boldsymbol{x})$ plays an important role. The relation between the expert and the resulting prototype is given qualitatively in Figure 1, where four cases (a–d) are identified. On one extreme, the expert is coarse, or simply absent, in which case, the optimization problem reduces to the maximization of the class probability function $p(\omega_c|\boldsymbol{x})$. On the other extreme, the expert is overfitted on some data distribution, and thus, the optimization problem becomes essentially the maximization of the expert $p(\boldsymbol{x})$ itself. When using AM for the purpose of model validation, an overfitted expert (case d) must be especially avoided, as the latter could hide interesting failure modes of the model $p(\omega_c|\boldsymbol{x})$. A slightly underfitted expert (case b), e.g. that simply favors images with natural colors,

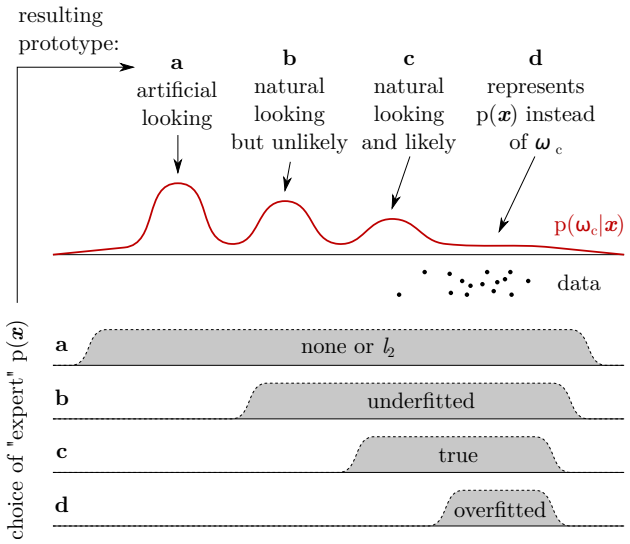


Figure 1: Cartoon illustrating how the expert $p(\boldsymbol{x})$ affects the prototype $\boldsymbol{x}^\star$ found by AM. The horizontal axis represents the input space, and the vertical axis represents the probability.

can already be sufficient. On the other hand, when using AM to gain knowledge on a correctly predicted concept $\omega_c$, the focus should be to prevent underfitting. Indeed, an underfitted expert would expose optima of $p(\omega_c|\boldsymbol{x})$ potentially distant from the data, and therefore, the prototype $\boldsymbol{x}^\star$ would not be truly representative of $\omega_c$.

### *3.3. Performing AM in Code Space*

In certain applications, data density models $p(\boldsymbol{x})$ can be hard to learn up to high accuracy, or very complex such that maximizing them becomes difficult. An alternative class of unsupervised models are generative models. They do not provide the density function directly, but are able to sample from it, usually via the following two steps:

1. Sample from a simple distribution $q(\boldsymbol{z}) \sim \mathcal{N}(0, I)$ defined in some abstract code space $\mathcal{Z}$.
2. Apply to the sample a decoding function $g : \mathcal{Z} \to \mathcal{X}$, that maps it back to the original input domain.

One such model is the generative adversarial network [19]. It learns a decoding function $g$ such that the generated data distribution is as hard as possible to discriminate from the true data distribution. The decoding function $g$ is learned in competition with a discriminant between the generated and the true distributions. The decoding function and the discriminant are typically chosen to be multilayer neural networks.

Nguyen et al. [37] proposed to build a prototype for $\omega_c$ by incorporating such generative model in the activation maximization framework. The optimization problem is redefined as:

$$\max_{\boldsymbol{z} \in \mathcal{Z}} \; \log p(\omega_c \,|\, g(\boldsymbol{z})) - \lambda \|\boldsymbol{z}\|^2,$$

where the first term is a composition of the newly introduced decoder and the original classifier, and where the second term is an $\ell_2$-norm regularizer in the code space. Once a solution $\boldsymbol{z}^\star$ to the optimization problem is found, the prototype for $\omega_c$ is obtained by decoding the solution, that is, $\boldsymbol{x}^\star = g(\boldsymbol{z}^\star)$. In Section 3.1, the $\ell_2$-norm regularizer in the input space was understood in the context of image data as favoring gray-looking images. The effect of the $\ell_2$-norm regularizer in the code space can instead be understood as encouraging codes that have high probability. Note however, that high probability codes do not necessarily map to high density regions of the input space.

To illustrate the qualitative differences between the methods of Sections 3.1–3.3, we consider the problem of interpreting MNIST classes as modeled by a three-layer DNN. We consider for this task (1) a simple $\ell_2$-norm regularizer $\lambda\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2$ where $\bar{\boldsymbol{x}}$ denotes the data mean for $\omega_c$, (2) a Gaussian RBM expert $p(\boldsymbol{x})$, and (3) a generative model with a two-layer decoding function, and the $\ell_2$-norm regularizer $\lambda\|\boldsymbol{z} - \bar{\boldsymbol{z}}\|^2$ where $\bar{\boldsymbol{z}}$ denotes the code mean for $\omega_c$. Corresponding architectures and found prototypes are shown in Figure 2. Each prototype is classified with full certainty by the DNN. However, only with an expert or a decoding function, the prototypes become sharp and realistic-looking.

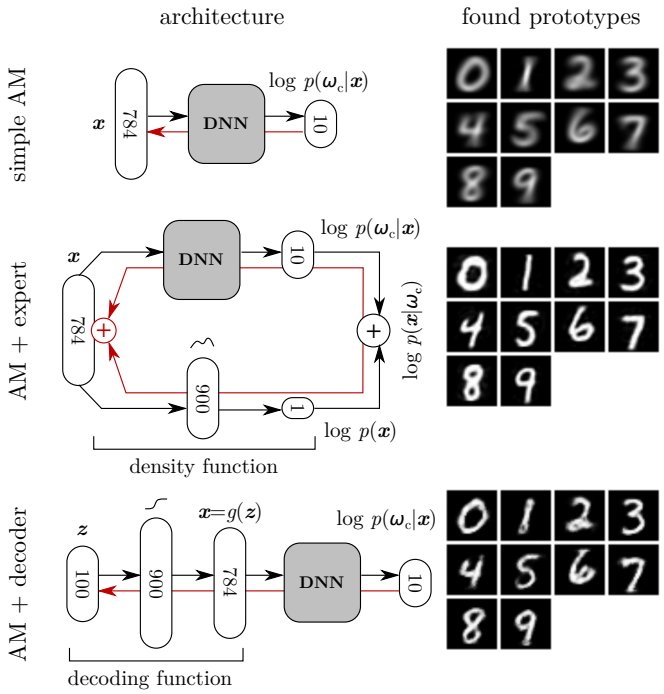


Figure 2: Architectures supporting AM procedures and found prototypes. Black arrows indicate the forward path and red arrows indicate the reverse path for gradient computation.

### *3.4. From Global to Local Analysis*

When considering complex machine learning problems, probability functions $p(\omega_c|\boldsymbol{x})$ and $p(\boldsymbol{x})$ might be multimodal or strongly elongated, so that no single prototype

$\boldsymbol{x}^\star$ fully represents the modeled concept $\omega_c$. The issue of multimodality is raised by Nguyen et al. [38], who demonstrate in the context of image classification, the benefit of interpreting a class $\omega_c$ using multiple local prototypes instead of a single global one.

Producing an exhaustive description of the modeled concept $\omega_c$ is however not always necessary. One might instead focus on a particular region of the input space. For example, biomedical data is best analyzed conditioned on a certain development stage of a medical condition, or in relation to a given subject or organ.

An expedient way of introducing locality into the analysis is to add a localization term $\eta \cdot \|\boldsymbol{x} - \boldsymbol{x}_0\|^2$ to the AM objective, where $\boldsymbol{x}_0$ is a reference point. The parameter $\eta$ controls the amount of localization. As this parameter increases, the question "*what is a good prototype of* $\omega_c$*?*" becomes however insubstantial, as the prototype $\boldsymbol{x}^\star$ converges to $\boldsymbol{x}_0$ and thus looses its information content.

Instead, when trying to interpret the concept $\omega_c$ locally, a more relevant question to ask is "*what features of* $\boldsymbol{x}$ *make it representative of the concept* $\omega_c$*?*". This question gives rise to a second type of analysis, that will be the focus of the rest of this tutorial.

## 4. Explaining DNN Decisions

In this section, we ask for a given data point $\boldsymbol{x}$, what makes it representative of a certain concept $\omega_c$ encoded in some output neuron of the deep neural network (DNN). The output neuron can be described as a function $f(\boldsymbol{x})$ of the input. A common approach is to view the data point $\boldsymbol{x}$ as a collection of features $(x_i)_{i=1}^d$, and to assign to each of these, a score $R_i$ determining how *relevant* the feature $x_i$ is for explaining $f(\boldsymbol{x})$. An example is given in Figure 3.

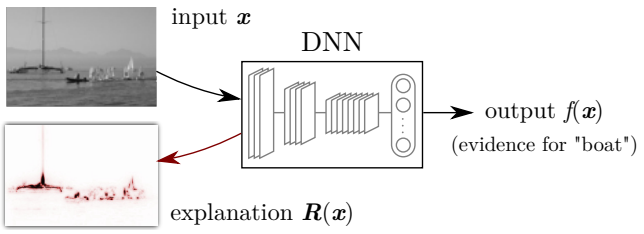


Figure 3: Explanation of the DNN prediction "boat" for an image $\boldsymbol{x}$ given as input.

In this example, an image is presented to a DNN and is classified as "boat". The prediction (encoded in the output layer) is then mapped back to the input domain. The explanation takes the form of a heatmap, where pixels with a high associated relevance score are shown in red.

### *4.1. Sensitivity Analysis*

A first approach to identify the most important input features is sensitivity analysis. It is based on the model's locally evaluated gradient or some other local measure of variation. A common formulation of sensitivity analysis defines relevance scores as

$$R_i(\boldsymbol{x}) = \Big(\frac{\partial f}{\partial x_i}\Big)^2,$$

where the gradient is evaluated at the data point $\boldsymbol{x}$. The most relevant input features are those to which the output is most sensitive. The technique is easy to implement for a deep neural network, since the gradient can be computed using backpropagation [11, 41].

Sensitivity analysis has been regularly used in scientific applications of machine learning such as medical diagnosis [25], ecological modeling [18], or mutagenicity prediction [6]. More recently, it was also used for explaining the classification of images by deep neural networks [44].

It is important to note, however, that sensitivity analysis does not produce an explanation of the function value $f(\boldsymbol{x})$ itself, but rather a *variation* of it. Sensitivity scores are indeed a decomposition of the local variation of the function as measured by the gradient square norm:

$$\textstyle\sum_{i=1}^d R_i(\boldsymbol{x}) = \|\nabla f(\boldsymbol{x})\|^2$$

Intuitively, when applying sensitivity analysis e.g. to a neural network detecting cars in images, we answer the question "*what makes this image more/less a car?*", rather than the more basic question "*what makes this image a car?*".

### *4.2. Simple Taylor Decomposition*

The Taylor decomposition [7, 5] is a method that explains the model's decision by decomposing the function value $f(\boldsymbol{x})$ as a sum of relevance scores. The relevance scores are obtained by identification of the terms of a first-order Taylor expansion of the function at some root point $\widetilde{\boldsymbol{x}}$ for which $f(\widetilde{\boldsymbol{x}}) = 0$. This expansion lets us rewrite the function as:

$$f(\boldsymbol{x}) = \textstyle\sum_{i=1}^d R_i(\boldsymbol{x}) + O(\boldsymbol{x}\boldsymbol{x}^\top)$$

where the relevance scores

$$R_i(\boldsymbol{x}) = \frac{\partial f}{\partial x_i}\Big|_{\boldsymbol{x}=\widetilde{\boldsymbol{x}}} \cdot (x_i - \widetilde{x}_i)$$

are the first-order terms, and where $O(\boldsymbol{x}\boldsymbol{x}^\top)$ contains all higher-order terms. Because these higher-order terms are typically non-zero, this analysis only provides a partial explanation of $f(\boldsymbol{x})$.

However, a special class of functions, piecewise linear and satisfying the property $f(t\,\boldsymbol{x}) = t\,f(\boldsymbol{x})$ for $t \geq 0$, is not subject to this limitation. Examples of such functions used in machine learning are homogeneous linear models, or deep ReLU networks (without biases). For these functions, we can always find a root point $\widetilde{\boldsymbol{x}} = \lim_{\varepsilon \to 0} \varepsilon \cdot \boldsymbol{x}$, that incidentally lies on the same linear region as the data point $\boldsymbol{x}$, and for which the second and higher-order terms are zero. In that case, the function can be rewritten as

$$f(\boldsymbol{x}) = \textstyle\sum_{i=1}^d R_i(\boldsymbol{x})$$

where the relevance scores simplify to

$$R_i(\boldsymbol{x}) = \frac{\partial f}{\partial x_i} \cdot x_i.$$

Relevance can here be understood as the product of sensitivity (given by the locally evaluated partial derivative) and saliency (given by the input value). That is, an input feature is relevant if it is both present in the data, and if the model reacts to it.

Later in this tutorial, we will also show how this simple technique serves as a primitive for building the more sophisticated deep Taylor decomposition [34].

### *4.3. Relevance Propagation*

An alternative way of decomposing the prediction of a DNN is to make explicit use of its feed-forward graph structure. The algorithm starts at the output of the network, and moves in the graph in reverse direction, progressively redistributing the prediction score (or total relevance) until the input is reached. The redistribution process must furthermore satisfy a local *relevance conservation* principle.

A physical analogy would be that of an electrical circuit where one injects a certain amount of current at the first endpoint, and measures the resulting current at the other endpoints. In this physical example, Kirchoff's conservation laws for current apply locally to each node of the circuit, but also ensure the conservation property at a global level.

The propagation approach was proposed by Landecker et al. [26] to explain the predictions of hierarchical networks, and was also introduced by Bach et al. [5] in the context of convolutional DNNs for explaining the predictions of these state-of-the-art models.

Let us consider a DNN where $j$ and $k$ are indices for neurons at two successive layers. Let $(R_k)_k$ be the relevance scores associated to neurons in the higher layer. We define $R_{j\leftarrow k}$ as the share of relevance that flows from neuron $k$ to neuron $j$. This share is determined based on the contribution of neuron $j$ to $R_k$, subject to the local relevance conservation constraint

$$\textstyle\sum_j R_{j\leftarrow k} = R_k.$$

The relevance of a neuron in the lower layer is then defined as the total relevance it receives from the higher layer:

$$R_j = \textstyle\sum_k R_{j\leftarrow k}$$

These two equations, when combined, ensure between all consecutive layers a relevance conservation property, which in turn also leads to a global conservation property from the neural network output to the input relevance scores:

$$\textstyle\sum_{i=1}^{d} R_i = \cdots = \sum_j R_j = \sum_k R_k = \cdots = f(\boldsymbol{x})$$

It should be noted that there are other explanation techniques that rely on the DNN graph structure, although not producing a decomposition of $f(\boldsymbol{x})$. Two examples are the *deconvolution* by Zeiler and Fergus [55], and *guided backprop* by Springenberg et al. [47]. They also work by applying a backward mapping through the graph, and generate interpretable patterns in the input domain, that are associated to a certain prediction or a feature map activation.

### *4.4. Practical Considerations*

Explanation techniques that derive from a decomposition principle provide several practical advantages: First, they give an implicit quantification of the share that can be imputed to individual input features. When the number of input variables is limited, the analysis can therefore be represented as a pie chart or histogram. If the number of input variables is too large, the decomposition can be coarsened by *pooling* relevance scores over groups of features.

For example, in RGB images, the three relevance scores of a pixel can be summed to obtain the relevance score of the whole pixel. The resulting pixel scores can be displayed as a heatmap. On an object recognition task, Lapuschkin et al. [27] further exploited this mechanism by pooling relevance over two large regions of the image: (1) the bounding box of the object to detect and (2) the rest of the image. This coarse analysis was used to quantify the reliance of the model on the object itself and on its spatial context.

In addition, when the explanation technique uses propagation in the model's graph, the quantity being propagated can be *filtered* to only include what flows through a certain neuron or feature map. This allows to capture individual components of an explanation, that would otherwise be entangled in the heatmap.

The pooling and filtering capabilities of each explanation technique are shown systematically in Table 1.

| | pooling | filtering |
|---|---|---|
| sensitivity analysis | ✓ | |
| simple Taylor | ✓ | |
| relevance propagation | ✓ | ✓ |
| *deconvolution* [55] | | ✓ |
| *guided backprop* [47] | | ✓ |

Table 1: Properties of various techniques for explaining DNN decisions. The first three entries correspond to the methods introduced in Sections 4.1–4.3.

## 5. The LRP Explanation Framework

In this section, we focus on the layer-wise relevance propagation (LRP) technique introduced by Bach et al. [5] for explaining deep neural network predictions. LRP is based on the propagation approach described in Section 4.3, and has been used in a number of practical applications, in particular, for model validation and analysis of scientific data. Some of these applications are discussed in Sections 8.1 and 8.2.

LRP is first described algorithmically in Section 5.1, and then shown in Section 5.2 to correspond in some cases to a deep Taylor decomposition of the model's decision [34]. Practical recommendations and tricks to make efficient use of LRP are then given in Section 6.

### 5.1. Propagation Rules for DNNs

In the original paper [5], LRP was applied to bag-of-words and deep neural network models. In this tutorial, we focus on the second type of models. Let the neurons of the DNN be described by the equation

$$a_k = \sigma\big(\textstyle\sum_j a_j w_{jk} + b_k\big),$$

with $a_k$ the neuron activation, $(a_j)_j$ the activations from the previous layer, and $w_{jk}, b_k$ the weight and bias parameters of the neuron. The function $\sigma$ is a positive and monotonically increasing activation function.

One propagation rule that fulfills local conservation properties, and that was shown to work well in practice is the $\alpha\beta$-rule given by:

$$R_j = \sum_k \Big(\alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-}\Big) R_k, \qquad (1)$$

where $()^+$ and $()^-$ denote the positive and negative parts respectively, and where the parameters $\alpha$ and $\beta$ are chosen subject to the constraints $\alpha - \beta = 1$ and $\beta \geq 0$. To avoid divisions by zero, small stabilizing terms can be introduced when necessary. The rule can be rewritten as

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k^\wedge + \sum_k \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-} R_k^\vee,$$

where $R_k^\wedge = \alpha R_k$ and $R_k^\vee = -\beta R_k$. It can now be interpreted as follows:

> *Relevance $R_k^\wedge$ should be redistributed to the lower-layer neurons $(a_j)_j$ in proportion to their excitatory effect on $a_k$. "Counter-relevance" $R_k^\vee$ should be redistributed to the lower-layer neurons $(a_j)_j$ in proportion to their inhibitory effect on $a_k$.*

Different combinations of parameters $\alpha, \beta$ were shown to modulate the qualitative behavior of the resulting explanation. As a naming convention, we denote, for example, by LRP-$\alpha_2\beta_1$, the fact of having chosen the parameters $\alpha = 2$ and $\beta = 1$ for this rule. In the context of image classification, a non-zero value for $\beta$ was shown empirically to have a sparsifying effect on the explanation [5, 34]. On the BVLC CaffeNet [24], LRP-$\alpha_2\beta_1$ was shown to work well, while for the deeper GoogleNet [50], LRP-$\alpha_1\beta_0$ was found to be more stable.

When choosing LRP-$\alpha_1\beta_0$, the propagation rule reduces to the simpler rule:

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k. \qquad (2)$$

The latter rule was also used by Zhang et al. [56] as part of an explanation method called excitation backprop.

### 5.2. LRP and Deep Taylor Decomposition

In this section, we show for deep ReLU networks a connection between LRP-$\alpha_1\beta_0$ and Taylor decomposition. We show in particular that when neurons are defined as

$$a_k = \max\big(0, \textstyle\sum_j a_j w_{jk} + b_k\big) \quad \text{with} \quad b_k \leq 0,$$

the application of LRP-$\alpha_1\beta_0$ at a given layer can be seen as computing a Taylor decomposition of the relevance at that layer onto the lower layer. The name "deep Taylor decomposition" then arises from the iterative application of Taylor decomposition from the top layer down to the input layer.

Let us assume that the relevance for the neuron $k$ can be written as $R_k = a_k c_k$, a product of the neuron activation $a_k$ and a term $c_k$ that is *constant* and *positive*. These two properties allow us to construct a "relevance neuron"

$$R_k = \max\big(0, \textstyle\sum_j a_j w'_{jk} + b'_k\big), \qquad (3)$$

with parameters $w'_{jk} = w_{jk} c_k$ and $b'_k = b_k c_k$. The relevance neuron is shown in Figure 4(a).

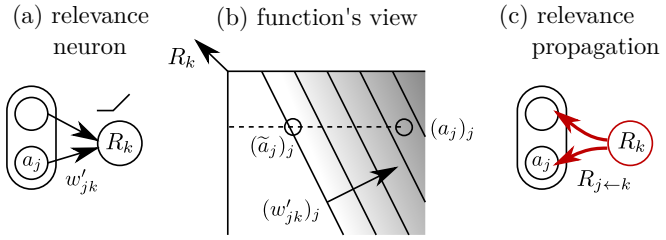


Figure 4: Diagram of the relevance neuron and its analysis. The root search domain is shown with a dashed line, and the relevance propagation resulting from decomposing $R_k$ is shown in red.

We now would like to propagate the relevance to the lower layer. For this, we perform a Taylor decomposition of $R_k$ on the lower-layer neurons. We search for the nearest root point $(\widetilde{a}_j)_j$ of $R_k$ on the segment $[(a_j 1_{w'_{jk}\leq 0})_j, (a_j)_j]$. The search strategy is visualized in Figure 4(b). Because the relevance neuron is piecewise linear, the Taylor expansion at the root point contains only first-order terms:

$$R_k = \sum_j \underbrace{\frac{\partial R_k}{\partial a_j}\Big|_{(\widetilde{a}_j)_j} \cdot (a_j - \widetilde{a}_j)}_{R_{j\leftarrow k}}$$

The first-order terms correspond to the decomposition of $R_k$ on the lower-layer neurons and have the closed-form expression

$$R_{j\leftarrow k} = \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k.$$

The resulting propagation of $R_k$ is shown in Figure 4(c). Summing $R_{j\leftarrow k}$ over all neurons $k$ to which neuron $j$ contributes yields exactly the LRP-$\alpha_1\beta_0$ propagation rule of Equation (2).

We now would like to verify that the procedure can be repeated one layer below. For this, we inspect the structure of $R_j$ and observe that it can be written as a product $R_j = a_j c_j$, where $a_j$ is the neuron activation and

$$\begin{aligned} c_j &= \sum_k \frac{w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k \\ &= \sum_k w_{jk}^+ \frac{\max\left(0, \sum_j a_j w_{jk} + b_k\right)}{\sum_j a_j w_{jk}^+} c_k \end{aligned}$$

is *positive* and also approximately *constant*. The latter property arises from the observation that the dependence of $c_j$ on the activation $a_j$ is only very indirect (diluted by two nested sums), and that the other terms $w_{jk}, w_{jk}^+, b_k, c_k$ are constant or approximately constant.

The positivity and near-constancy of $c_j$ implies that similar relevance neuron to the one of Equation (3) can be built for neuron $j$, for the purpose of redistributing relevance on the layer before. The decomposition process can therefore be repeated in the lower layers, until the first layer of the neural network is reached, thus, performing a deep Taylor decomposition [34].

In the derivation above, the segment on which we search for a root point incidentally guarantees (1) membership of the root point to the domain of ReLU activations and (2) positivity of relevance scores. These guarantees can also be brought to other types of layers (e.g. input layers receiving real values or pixels intensities), by searching for a root point $(\widetilde{a}_j)_j$ on a different segment. This leads to different propagation rules, some of which are listed in Table 2. Details on how to derive these rules are given in the original paper [34]. We refer to these rules as "deep Taylor LRP" rules.

| Input domain | Rule |
|---|---|
| ReLU activations ($a_j \geq 0$) | $R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k$ |
| Pixel intensities ($x_i \in [l_i, h_i]$, $l_i \leq 0 \leq h_i$) | $R_i = \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j$ |
| Real values ($x_i \in \mathbb{R}$) | $R_i = \sum_j \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j$ |

Table 2: Deep Taylor LRP rules derived for various layer types. The first rule applies to the hidden layers, and the next two rules apply to the first layer.

### *5.3. Handling Special Layers*

Practical neural networks are often composed of special layers, for example, $\ell_p$-pooling layers (including sum-pooling and max-pooling as the two extreme cases), and normalization layers. The original paper by Bach et al. [5] uses a winner-take-all redistribution policy for max-pooling layers, where all relevance goes to the most activated neuron in the pool. Instead, Montavon et al. [34] recommend to apply for $\ell_p$-pooling layers the following propagation rule:

$$R_j = \frac{x_j}{\sum_j x_j} R_k,$$

i.e. redistribution is proportional to neuron activations in the pool. This redistribution rule ensures explanation continuity (see Section 7.1 for an introduction to this concept).

With respect to normalization layers, Bach et al. [5] proposed to ignore them in the relevance propagation pass. Alternately, Binder et al. [10] proposed for these layers a more sophisticated rule based on a local Taylor expansion of the normalization function, with some benefits in terms of explanation selectivity.

## 6. Recommendations and Tricks for LRP

Machine learning methods are often described in papers at an abstract level, for maximum generality. However, a good choice of hyperparameters is usually necessary to make them work well on real-world problems, and tricks are often used to make most efficient use of these methods and extend their capabilities [8, 23, 35]. Likewise, the LRP framework introduced in Section 5, also comes with a list of recommendations and tricks, some of which are given below.

### *6.1. How to Choose the Model to Explain*

The LRP approach is aimed at general feedforward computational graphs. However, it was most thoroughly studied, both theoretically [34] and empirically [42], on specific types of models such as convolutional neural networks with ReLU nonlinearities. This leads to our first recommendation:

> *Apply LRP to classes of models where it was successfully applied in the past. In absence of trained model of such class, consider training your own.*

We have also observed empirically that in order for LRP to produce good explanations, the number of fully connected layers should be kept low, as LRP tends for these layers to redistribute relevance to too many lower-layer neurons, and thus, loose selectivity.

> *As a first try, consider a convolutional ReLU network, as deep as needed, but with not too many fully connected layers. Use dropout* [48] *in these layers.*

For the LRP procedure to best match the deep Taylor decomposition framework outlined in Section 5.2, sum-pooling or average-pooling layers should be preferred to max-pooling layers, and bias parameters of the network should either be zero or negative.

> *Prefer sum-pooling to max-pooling, and force biases to be zero or negative at training time.*

Negative biases will contribute to further sparsify the network activations, and therefore, also to better disentangle the relevance at each layer.

### *6.2. How to Choose the LRP Rules for Explanation*

In presence of a deep neural network that follows the recommendations above, a first set of propagation rules to be tried are the deep Taylor LRP rules of Table 2, which exhibit a stable behavior, and that are also well understood theoretically. These rules produce for positive predictions a positive heatmap, where input variables are deemed relevant if $R_i > 0$ or irrelevant if $R_i = 0$.

> *As a default choice for relevance propagation, use the deep Taylor LRP rules given in Table 2.*

In presence of predictive uncertainty, a certain number of input variables might be in contradiction with the prediction, and the concept of "negative relevance" must therefore be introduced. Negative relevance can be injected into the explanation in a controlled manner by setting the coefficients of the $\alpha\beta$-rule of Equation (1) to an appropriate value.

> *If negative relevance is needed, or the heatmaps are too diffuse, replace the rule LRP-$\alpha_1\beta_0$ by LRP-$\alpha_2\beta_1$ in the hidden layers.*

The LRP-$\alpha_1\beta_0$ and LRP-$\alpha_2\beta_1$ rules were shown to work well on image classification [34], but there is a potentially much larger set of rules that we can choose from. For example, the "$\epsilon$-rule" [5] was applied successfully to text categorization [3, 4]. To choose the most appropriate rule among the set of possible ones, a good approach is to define a heatmap quality criterion, and select the rule at each layer accordingly. One such quality criterion called "pixel-flipping" measures heatmap selectivity and is later introduced in Section 7.2.

> *If the heatmaps obtained with LRP-$\alpha_1\beta_0$ and LRP-$\alpha_2\beta_1$ are unsatisfactory, consider a larger set of propagation rules, and use pixel-flipping to select the best one.*

### *6.3. Tricks for Implementing LRP*

Let us consider the LRP-$\alpha_1\beta_0$ propagation rule of Equation (2):

$$R_j = a_j \sum_k \frac{w^+_{jk}}{\sum_j a_j w^+_{jk}} R_k,$$

where we have for convenience moved the neuron activation $a_j$ outside the sum. This rule can be written as four elementary computations, all of which can also expressed in vector form:

| element-wise | vector form | |
|---|---|---|
| $z_k \leftarrow \sum_j a_j w^+_{jk}$ | $\boldsymbol{z} \leftarrow W_+^\top \cdot \boldsymbol{a}$ | (4) |
| $s_k \leftarrow R_k / z_k$ | $\boldsymbol{s} \leftarrow \boldsymbol{R} \oslash \boldsymbol{z}$ | (5) |
| $c_j \leftarrow \sum_k w^+_{jk} s_k$ | $\boldsymbol{c} \leftarrow W_+ \cdot \boldsymbol{s}$ | (6) |
| $R_j \leftarrow a_j c_j$ | $\boldsymbol{R} \leftarrow \boldsymbol{a} \odot \boldsymbol{c}$ | (7) |

In the vector form computations, $\oslash$ and $\odot$ denote the element-wise division and multiplication. The variable $W$ denotes the weight matrix connecting the neurons of the two consecutive layers, and $W_+$ is the matrix retaining only the positive weights of $W$ and setting remaining weights to zero. This vector form is useful to implement LRP for fully connected layers.

In convolution layers, the matrix-vector multiplications of Equations (4) and (6) can be more efficiently implemented by borrowing the `forward` and `backward` methods used for forward activation and gradient propagation. These methods are readily available in many neural network libraries and are typically highly optimized. Based on these high-level primitives, LRP can implemented by the following sequence of operations:

```
def lrp(layer,a,R):

    clone = layer.clone()
    clone.W = maximum(0,layer.W)
    clone.B = 0

    z = clone.forward(a)
    s = R / z
    c = clone.backward(s)

    return a * c
```

The function `lrp` receives as arguments the `layer` through which the relevance should be propagated, the activations "`a`" at the layer input, and the relevance scores "`R`" at the layer output. The function returns the redistributed relevance at the layer input. Sample code is provided at `http://heatmapping.org/tutorial`. This modular approach was also used by Zhang et al. [56] to implement the excitation backprop method.

### *6.4. Translation Trick for Denoising Heatmaps*

It is sometimes observed that, for classifiers that are not optimally trained or structured, LRP heatmaps have unaesthetic features. This can be caused, for example, by the presence of noisy first-layer filters, or a large stride parameter in the first convolution layer. These effects can be mitigated by considering the explanation not of a single input image but the explanations of multiple slightly translated versions of the image. The heatmaps for these translated versions are then recombined by applying to them the inverse translation operation and averaging them up.

In mathematical terms, the improved heatmap is given by:

$$\boldsymbol{R}^\star(\boldsymbol{x}) = \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}} \tau^{-1}(\boldsymbol{R}(\tau(\boldsymbol{x})))$$

where $\tau, \tau^{-1}$ denote the translation and its inverse, and $\mathcal{T}$ is the set of all translations of a few pixels.

### *6.5. Sliding Window Explanations for Large Images*

In applications such as medical imaging or scene parsing, the images to be processed are typically larger than the what the neural network receives as input. Let $\boldsymbol{X}$ be this large image. The LRP procedure can be extended for this scenario by applying a sliding window strategy, where the neural network is moved through the whole image, and where heatmaps produced at various locations must then be combined into a single large heatmap. Technically, we define the quantity to explain as:

$$g(\boldsymbol{X}) = \sum_{s\in\mathcal{S}} f(\underbrace{\boldsymbol{X}[s]}_{\boldsymbol{x}})$$

where $\boldsymbol{X}[s]$ extracts a patch from the image $\boldsymbol{X}$ at location $s$, and $\mathcal{S}$ is the set of all locations in that image. Pixels then receive relevance from all patches to which they belong and in which they contribute to the function value $f(\boldsymbol{x})$. This technique is illustrated in Figure 5.

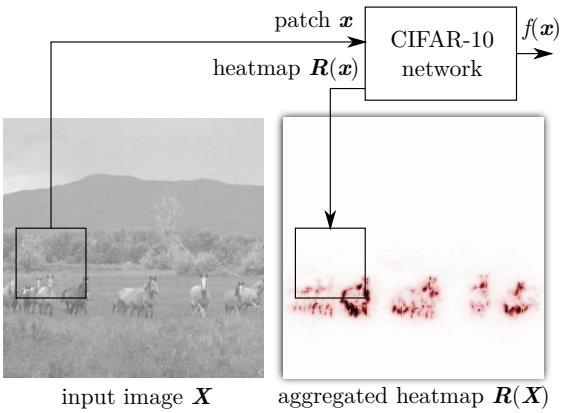


Figure 5: Highlighting in a large image pixels that are relevant for the CIFAR-10 class "horse", using the sliding window technique.

The convolutional neural network is a special case that can technically receive an input of any size. A heatmap can be obtained directly from it by redistributing the top-layer activations using LRP. This direct approach can provide a computational gain compared to the sliding window approach. However, it is not strictly equivalent and can produce unreliable heatmaps, e.g. when the network uses border-padded convolutions. If in doubt, it is preferable to use the sliding window formulation.

### *6.6. Visualize Relevant Pattern*

Due to their characteristic spatial structure, LRP heatmaps readily provide intuition on which input pattern the model has used to make its prediction. However, in presence of cluttered scenes, a better visualization can be obtained by using the heatmap as a mask to extract relevant pixels (and colors) from the image. We call the result of the masking operation the *pattern* $\boldsymbol{P}(\boldsymbol{x})$ that we compute as:

$$\boldsymbol{P}(\boldsymbol{x}) = \boldsymbol{x} \odot \boldsymbol{R}(\boldsymbol{x}).$$

Here, we assume that the heatmap scores have been preliminarily normalized between 0 and 1 through rescaling and/or clipping so that the masked image remains in the original color space. This visualization of LRP heatmaps makes it also more directly comparable to the visualizations techniques proposed in [55, 47].

## 7. Quantifying Explanation Quality

In Sections 4 and 5, we have introduced a number of explanation techniques. While each technique is based on its own intuition or mathematical principle, it is also important to define at a more abstract level what are the characteristics of a good explanation, and to be able to test for these characteristics quantitatively. A quantitative framework allows to compare explanation techniques specifically for a target problem, e.g. ILSVRC or MIT Places [42]. We present in Sections 7.1 and 7.2 two important properties of an explanation, along with possible evaluation metrics.

### *7.1. Explanation Continuity*

A first desirable property of an explanation technique is that it produces a continuous explanation function. Here, we implicitly assume that the prediction function $f(\boldsymbol{x})$ is also continuous. We would like to ensure in particular the following behavior:

> *If two data points are nearly equivalent, then the explanations of their predictions should also be nearly equivalent.*

Explanation continuity (or lack of it) can be quantified by looking for the strongest variation of the explanation $\boldsymbol{R}(\boldsymbol{x})$ in the input domain:

$$\max_{\boldsymbol{x}\neq\boldsymbol{x}'} \frac{\|\boldsymbol{R}(\boldsymbol{x}) - \boldsymbol{R}(\boldsymbol{x}')\|_1}{\|\boldsymbol{x} - \boldsymbol{x}'\|_2}.$$

When $f(\boldsymbol{x})$ is a deep ReLU network, both sensitivity analysis and simple Taylor decomposition have sharp discontinuities in their explanation function. On the other hand, deep Taylor LRP produces continuous explanations. This is illustrated in Figure 6 for the simple function $f(\boldsymbol{x}) = \max(x_1, x_2)$ in $\mathbb{R}^2_+$, here implemented by the two-layer ReLU network

$$\begin{aligned} f(\boldsymbol{x}) = \max\big(0\,, & 0.5\max(0, x_1 - x_2) \\ & + 0.5\max(0, x_2 - x_1) \\ & + 0.5\max(0, x_1 + x_2)\big). \end{aligned}$$

It can be observed that despite the continuity of the prediction function, the explanations offered by sensitivity

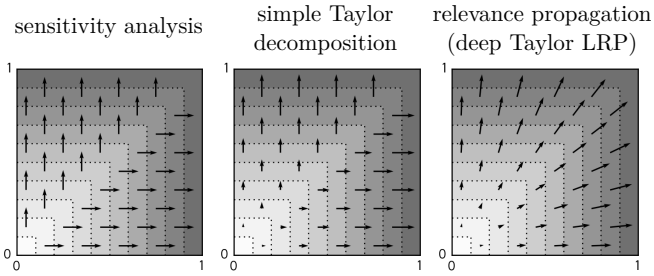


Figure 6: Explaining $\max(x_1, x_2)$. Function values are represented as a contour plot, with dark regions corresponding to high values. Relevance scores are represented as a vector field, where horizontal and vertical components are the relevance of respective input variables.

analysis and simple Taylor decomposition are discontinuous on the line $x_1 = x_2$. Here, only deep Taylor LRP produces a smooth transition.

More generally, techniques that rely on the function's gradient, such as sensitivity analysis or simple Taylor decomposition, are more exposed to the derivative noise [45] that characterizes complex machine learning models. Consequently, these techniques are also unlikely to score well in terms of explanation continuity.

Figure 7 shows the function value and the relevance scores for each technique, when applying them to a convolutional DNN trained on MNIST. Although the function itself is relatively low-varying, strong variations occur in the explanations. Here again, only deep Taylor LRP produces reasonably continuous explanations.

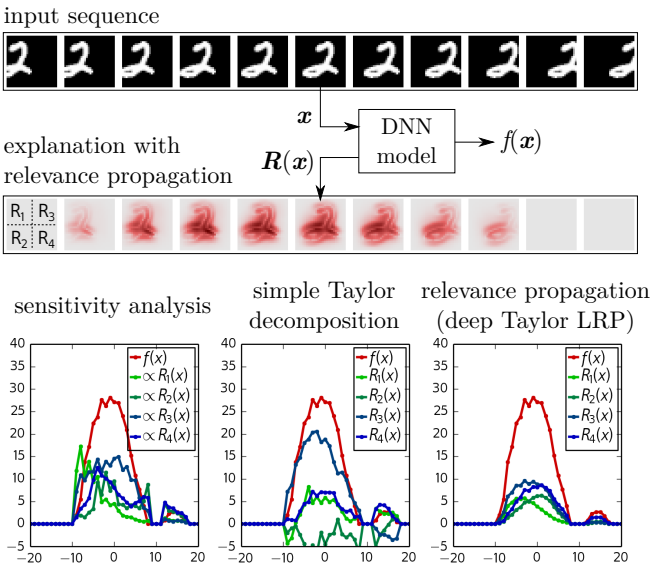


Figure 7: Classification "2" by a DNN, explained by different methods, as we move a handwritten digit from left to right in its receptive field. Relevance scores are pooled into four quadrants, and are tracked as we apply the translation operation.

### 7.2. *Explanation Selectivity*

Another desirable property of an explanation is that it redistributes relevance to variables that have the strongest impact on the function $f(\boldsymbol{x})$. Bach et al. [5] and Samek et al. [42] proposed to quantify selectivity by measuring how fast $f(\boldsymbol{x})$ goes down when removing features with highest relevance scores.

The method was introduced for image data under the name "pixel-flipping" [5, 42], and was also adapted to text data, where words selected for removal have their word embeddings set to zero [3]. The method works as follows:

> **repeat** until all features have been removed:
>
> - record the current function value $f(\boldsymbol{x})$
> - find feature $i$ with highest relevance $R_i(\boldsymbol{x})$
> - remove that feature ($\boldsymbol{x} \leftarrow \boldsymbol{x} - \{x_i\}$)
>
> make a plot with all recorded function values, and return the area under the curve (AUC) for that plot.

A sharp drop of function's value, characterized by a low AUC score indicates that the correct features have been identified as relevant. AUC results can be averaged over a large number of examples in the dataset.

Figure 8 illustrates the procedure on the same DNN as in Figure 7. At each iteration, a patch of size $4 \times 4$ corresponding to the region with highest relevance is set to black. The plot on the right keeps track of the function score as the features are being progressively removed. In this particular case, the plot indicates that deep Taylor LRP is more selective than sensitivity analysis and simple Taylor decomposition.

It is important to note however, that the result of the analysis depends to some extent on the feature removal process. Various feature removal strategies can be used, but a general rule is that it should keep as much as possible the image being modified on the data manifold. Indeed,

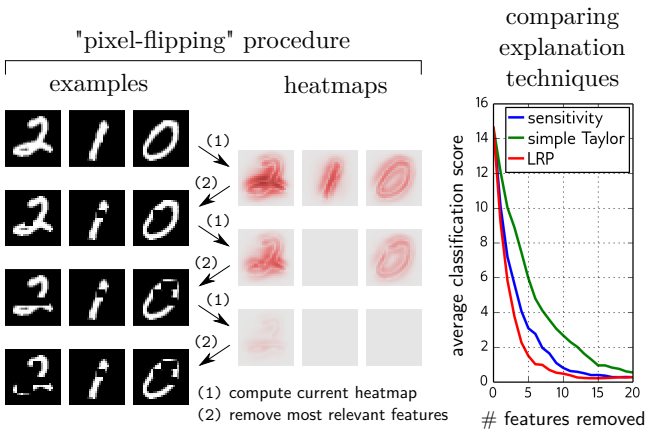


Figure 8: Illustration of the "pixel-flipping" procedure. At each step, the heatmap is used to determine which region to remove (by setting it to black), and the classification score is recorded.

this guarantees that the DNN continues to work reliably through the whole feature removal procedure. This in turn makes the analysis less subject to uncontrolled factors of variation.

## 8. Applications

Potential applications of explanation techniques are vast and include as diverse domains as extraction of domain knowledge, computer-assisted decisions, data filtering, or compliance. We focus in this section on two types of applications: validation of a trained model, and analysis of scientific data.

### *8.1. Model Validation*

Model validation is usually achieved by measuring the error on some validation set disjoint from the training data. While providing a simple way to compare different machine learning models in practice, it should be reminded that the validation error is only a proxy for the true error and that the data distribution and labeling process might differ. A human inspection of the model rendered interpretable can be a good complement to the validation procedure. We present two recent examples showing how explainability allows to better validate a machine learning model by pointing out at some unsuspected qualitative properties of it.

Arras et al. [3] considered a document classification task on the 20-Newsgroup dataset, and compared the explanations of a convolutional neural network (CNN) trained on word2vec inputs to the explanations of a support vector machine (SVM) trained on bag-of-words (BoW) document representations. They observed that, although both models produce a similar test error, the CNN model assigns most relevance to a small number of keywords, whereas the SVM classifier relies on word count regularities. Figure 9(a) displays explanations for an example of the target class `sci.space`.

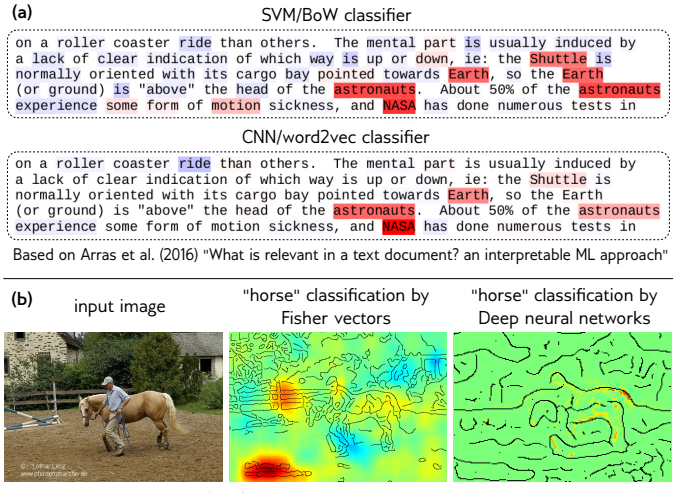


Figure 9: Examples taken from the literature of model validation via explanation. (a) Explanation of the concept "`sci.space`" by two text classifiers. (b) Unexpected use of copyright tags by the Fisher vector model for predicting the class "horse".

Lapuschkin et al. [27] compared the decisions taken by convolutional DNN transferred from ImageNet, and a Fisher vector classifier on PASCAL VOC 2012 images. Although both models reach similar classification accuracy on the category "horse", the authors observed that they use different strategies to classify images of that category. Explanations for a given image are shown in Figure 9(b). The deep neural network looks at the contour of the actual horse, whereas the Fisher vector model (of more rudimentary structure and trained with less data) relies mostly on a copyright tag, that happens to be present on many horse images. Removing the copyright tag in the test images would consequently significantly decrease the measured accuracy of the Fisher vector model but leave the deep neural network predictions unaffected.

### *8.2. Analysis of Scientific Data*

Beyond model validation, techniques of explanation can also be applied to shed light on scientific problems where human intuition and domain knowledge is often limited. Simple statistical tests and linear models have proved useful to identify correlations between different variables of a system, however, the measured correlations typically remain weak due to the inability of these models to capture the underlying complexity and nonlinearity of the studied problem. For a long time, the computational scientist would face a tradeoff between interpretability and predictive power, where linear models would sometimes be preferred to nonlinear models despite their lower predictive power. We give below a selection of recent works in various fields of research, that combine deep neural networks and explanation techniques to extract insight on the studied scientific problems.

In the domain of atomistic simulations, powerful machine learning models have been produced to link molecular structure to electronic properties [36, 21, 43, 16]. These models have been trained in a data-driven manner, without simulated physics involved into the prediction. In particular, Schütt et al. [43] proposed a deep tensor neural network model that incorporates sufficient structure and representational power to simultaneously achieve high predictive power and explainability. Using a test-charge perturbation analysis (a variant of sensitivity analysis where one measures the effect on the neural network output of inserting a charge at a given location), three-dimensional response maps were produced that highlight for each individual molecule spatial structures that were the most relevant for explaining the modeled structure-property relationship. Example of response maps are given in Figure 10(a) for various molecules.

Sturm et al. [49] showed that explanation techniques can also be applied to EEG brain recording data. Because the input EEG pattern can take different forms (due to different users, environments, or calibration of the acquisition

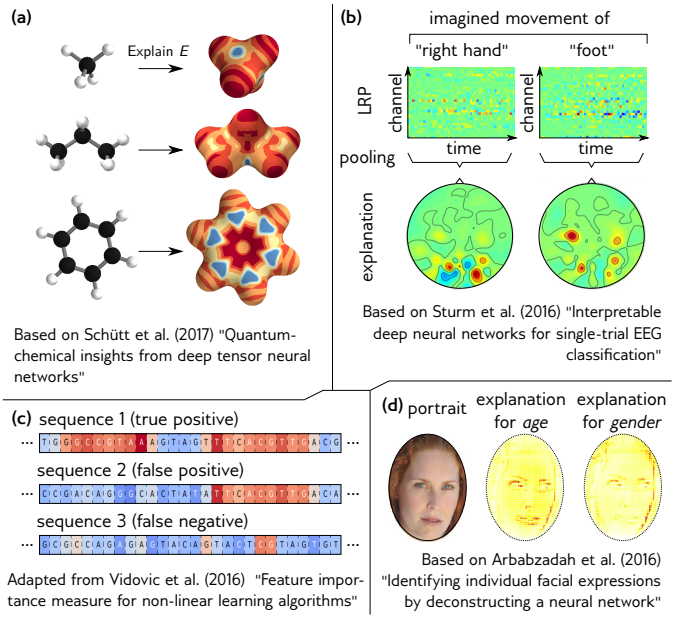


Figure 10: Overview of several applications of machine learning explanation techniques in the sciences. (a) Molecular response maps for quantum chemistry, (b) EEG heatmaps for neuroimaging, (c) extracting relevant information from gene sequences, (d) analysis of facial appearance.

device), it is important to produce an individual explanation that adapts to these parameters. After training a neural network to map EEG patterns to a set of movements imagined by the user ("right hand" and "foot"), a LRP decomposition of that prediction could be achieved in the EEG input domain (a spatiotemporal signal capturing the electrode measurements at various positions on the skull and at multiple time steps), and pooled temporally to produce EEG heatmaps revealing from which part of the brain the decision for "right hand" or "foot" originates. An interesting property of decomposition techniques in this context is that temporally pooling preserves the total function value, and thus, still corresponds to a decomposition of the prediction. Example of these individual EEG brain maps are given in Figure 10(b). For classical linear explanation of neural activation patterns in cognitive brain science experiments or Brain Computer Interfacing, see [13, 30, 12, 22].

Deep neural networks have also been proposed to make sense of the human genome. Alipanahi et al. [1] trained a convolutional neural network to map the DNA sequence to protein binding sites. In a second step, they asked what are the nucleotides of that sequence that are the most relevant for explaining the presence of these binding sites. For this, they used a perturbation-based analysis, similar to the sensitivity analysis described in Section 4.1, where the relevance score of each nucleotide is measured based on the effect of mutating it on the neural network prediction. Other measures of feature importance for individual gene sequences have been proposed [53] that apply to a broad class of nonlinear models, from deep networks to weighted degree kernel classifiers. Examples of heatmaps representing relevant genes for various sequences and prediction outcomes are shown in Figure 10(c).

Explanation techniques also have a potential application in the analysis of face images. These images may reveal a wide range of information about the person's identity, emotional state, or health. However, interpreting them directly in terms of actual features of the input image can be difficult. Arbabzadah et al. [2] applied a LRP technique to identify which pixels in a given image are responsible for explaining, for example, the age and gender attributes. Example of pixel-wise explanations are shown in Figure 10(d).

## 9. Conclusion

Building transparent machine learning systems is a convergent approach to both extracting novel domain knowledge and performing model validation. As machine learning is increasingly used in real-world decision processes, the necessity for transparent machine learning will continue to grow. Examples that illustrate the limitations of black-box methods were mentioned in Section 8.1.

This tutorial has covered two key directions for improving machine learning transparency: *interpreting* the concepts learned by a model by building prototypes, and *explaining* of the model's decisions by identifying the relevant input variables. The discussion mainly abstracted from the exact choice of deep neural network, training procedure, or application domain. Instead, we have focused on the more conceptual developments, and connected them to recent practical successes reported in the literature.

In particular we have discussed the effect of linking prototypes to the data, via a data density function or a generative model. We have described the crucial difference between sensitivity analysis and decomposition in terms of what these analyses seek to explain. Finally, we have outlined the benefit in terms of robustness, of treating the explanation problem with graph propagation techniques rather than with standard analysis techniques.

This tutorial has focused on post-hoc interpretability, where we do not have full control over the model's structure. Instead, the techniques of interpretation should apply to a general class of nonlinear machine learning models, no matter how they were trained and who trained them – even fully trained models that are available for download like BVLC CaffeNet [24] or GoogleNet [50]

In that sense the presented novel technological development in ML allowing for interpretability is an orthogonal strand of research independent of new developments for improving neural network models and their learning algorithms. We would like to stress that all new developments can in this sense always profit in addition from interpretability.

## Acknowledgments

We gratefully acknowledge discussions and comments on the manuscript by our colleagues Sebastian Lapuschkin, and Alexander Binder. This work was supported by the Brain Korea 21 Plus Program through the National Research Foundation of Korea; the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government [No. 2017-0-00451]; the Deutsche Forschungsgemeinschaft (DFG) [grant MU 987/17-1]; and the German Ministry for Education and Research as Berlin Big Data Center (BBDC) [01IS14013A]. This publication only reflects the authors views. Funding agencies are not liable for any use that may be made of the information contained herein.

## References

[1] Alipanahi, B., Delong, A., Weirauch, M. T., Frey, B. J., jul 2015. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nature Biotechnology 33 (8), 831–838.

[2] Arbabzadah, F., Montavon, G., Müller, K.-R., Samek, W., 2016. Identifying individual facial expressions by deconstructing a neural network. In: Pattern Recognition - 38th German Conference, GCPR 2016, Hannover, Germany, September 12-15, 2016, Proceedings. pp. 344–354.

[3] Arras, L., Horn, F., Montavon, G., Müller, K.-R., Samek, W., 2016. "What is relevant in a text document?": An interpretable machine learning approach. CoRR abs/1612.07843.

[4] Arras, L., Montavon, G., Müller, K.-R., Samek, W., 2017. Explaining recurrent neural network predictions in sentiment analysis. CoRR abs/1706.07206.

[5] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., Samek, W., 07 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE 10 (7), 1–46.

[6] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.-R., 2010. How to explain individual classification decisions. Journal of Machine Learning Research 11, 1803–1831.

[7] Bazen, S., Joutard, X., 2013. The Taylor decomposition: A unified generalization of the Oaxaca method to nonlinear models. Working papers, HAL.

[8] Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade - Second Edition. pp. 437–478.

[9] Berkes, P., Wiskott, L., 2006. On the analysis and interpretation of inhomogeneous quadratic forms as receptive fields. Neural Computation 18 (8), 1868–1895.

[10] Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., Samek, W., 2016. Layer-wise relevance propagation for neural networks with local renormalization layers. In: Artificial Neural Networks and Machine Learning - ICANN 2016 - 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II. pp. 63–71.

[11] Bishop, C. M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA.

[12] Blankertz, B., Lemm, S., Treder, M. S., Haufe, S., Müller, K.-R., 2011. Single-trial analysis and classification of ERP components - A tutorial. NeuroImage 56 (2), 814–825.

[13] Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., Müller, K.-R., 2008. Optimizing spatial filters for robust EEG single-trial analysis. IEEE Signal Processing Magazine 25 (1), 41–56.

[14] Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L. D., Muller, U., 2017. Explaining how a deep neural network trained with end-to-end learning steers a car. CoRR abs/1704.07911.

[15] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N., 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015. pp. 1721–1730.

[16] Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schütt, K. T., Müller, K.-R., may 2017. Machine learning of accurate energy-conserving molecular force fields. Science Advances 3 (5), e1603015.

[17] Erhan, D., Bengio, Y., Courville, A., Vincent, P., Jun. 2009. Visualizing higher-layer features of a deep network. Tech. Rep. 1341, University of Montreal, also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.

[18] Gevrey, M., Dimopoulos, I., Lek, S., feb 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological Modelling 160 (3), 249–264.

[19] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., Bengio, Y., 2014. Generative adversarial nets. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 2672–2680.

[20] Hansen, K., Baehrens, D., Schroeter, T., Rupp, M., Müller, K.-R., sep 2011. Visual interpretation of kernel-based prediction models. Molecular Informatics 30 (9), 817–826.

[21] Hansen, K., Biegler, F., Ramakrishnan, R., Pronobis, W., von Lilienfeld, O. A., Müller, K.-R., Tkatchenko, A., jun 2015. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. The Journal of Physical Chemistry Letters 6 (12), 2326–2331.

[22] Haufe, S., Meinecke, F. C., Görgen, K., Dähne, S., Haynes, J.-D., Blankertz, B., Bießmann, F., 2014. On the interpretation of weight vectors of linear models in multivariate neuroimaging. NeuroImage 87, 96–110.

[23] Hinton, G. E., 2012. A practical guide to training restricted Boltzmann machines. In: Neural Networks: Tricks of the Trade - Second Edition. pp. 599–619.

[24] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia, MM'14, Orlando, FL, USA, November 03 - 07, 2014. pp. 675–678.

[25] Khan, J., Wei, J. S., Ringnér, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., Meltzer, P. S., jun 2001. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine 7 (6), 673–679.

[26] Landecker, W., Thomure, M. D., Bettencourt, L. M. A., Mitchell, M., Kenyon, G. T., Brumby, S. P., 2013. Interpreting individual classifications of hierarchical networks. In: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013. pp. 32–38.

[27] Lapuschkin, S., Binder, A., Montavon, G., Müller, K.-R., Samek, W., 2016. Analyzing classifiers: Fisher vectors and deep neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 2912–2920.

[28] Lee, H., Grosse, R. B., Ranganath, R., Ng, A. Y., 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009. pp. 609–616.

[29] Leek, J. T., Scharpf, R. B., Bravo, H. C., Simcha, D., Langmead, B., Johnson, W. E., Geman, D., Baggerly, K., Irizarry, R. A., sep 2010. Tackling the widespread and critical impact of batch effects in high-throughput data. Nature Reviews Genetics 11 (10), 733–739.

[30] Lemm, S., Blankertz, B., Dickhaus, T., Müller, K.-R., 2011.

Introduction to machine learning for brain imaging. NeuroImage 56 (2), 387–399.
[31] Li, J., Chen, X., Hovy, E. H., Jurafsky, D., 2016. Visualizing and understanding neural models in NLP. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016. pp. 681–691.
[32] Lipton, Z. C., 2016. The mythos of model interpretability. CoRR abs/1606.03490.
[33] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 3111–3119.
[34] Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.-R., 2017. Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognition 65, 211–222.
[35] Montavon, G., Orr, G., Müller, K.-R., 2012. Neural Networks: Tricks of the Trade, 2nd Edition. Springer Publishing Company, Inc.
[36] Montavon, G., Rupp, M., Gobre, V., Vazquez-Mayagoitia, A., Hansen, K., Tkatchenko, A., Müller, K.-R., von Lilienfeld, O. A., sep 2013. Machine learning of molecular electronic properties in chemical compound space. New Journal of Physics 15 (9), 095003.
[37] Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J., 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 3387–3395.
[38] Nguyen, A., Yosinski, J., Clune, J., 2016. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. CoRR abs/1602.03616.
[39] Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D. S., Fyshe, A., Pearcy, B., Macdonell, C., Anvik, J., 2006. Visual explanation of evidence with additive classifiers. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA. pp. 1822–1829.
[40] Ribeiro, M. T., Singh, S., Guestrin, C., 2016. "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, San Francisco, CA, USA, August 13-17, 2016. pp. 1135–1144.
[41] Rumelhart, D. E., Hinton, G. E., Williams, R. J., oct 1986. Learning representations by back-propagating errors. Nature 323 (6088), 533–536.
[42] Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., 2016. Evaluating the visualization of what a deep neural network has learned. IEEE Transactions on Neural Networks and Learning Systems, 1–14.
[43] Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., Tkatchenko, A., jan 2017. Quantum-chemical insights from deep tensor neural networks. Nature Communications 8, 13890.
[44] Simonyan, K., Vedaldi, A., Zisserman, A., 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR abs/1312.6034.
[45] Snyder, J. C., Rupp, M., Hansen, K., Müller, K.-R., Burke, K., jun 2012. Finding density functionals with machine learning. Physical Review Letters 108 (25).
[46] Soneson, C., Gerster, S., Delorenzi, M., 06 2014. Batch effect confounding leads to strong bias in performance estimates obtained by cross-validation. PLOS ONE 9 (6), 1–13.
[47] Springenberg, J. T., Dosovitskiy, A., Brox, T., Riedmiller, M. A., 2014. Striving for simplicity: The all convolutional net. CoRR abs/1412.6806.
[48] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15 (1), 1929–1958.
[49] Sturm, I., Lapuschkin, S., Samek, W., Müller, K.-R., dec 2016. Interpretable deep neural networks for single-trial EEG classification. Journal of Neuroscience Methods 274, 141–145.
[50] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1–9.
[51] Taylor, B. J., 2005. Methods and Procedures for the Verification and Validation of Artificial Neural Networks. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
[52] van den Oord, A., Kalchbrenner, N., Kavukcuoglu, K., 2016. Pixel recurrent neural networks. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. pp. 1747–1756.
[53] Vidovic, M. M.-C., Görnitz, N., Müller, K.-R., Kloft, M., 2016. Feature importance measure for non-linear learning algorithms. CoRR abs/1611.07567.
[54] Vidovic, M. M.-C., Kloft, M., Müller, K.-R., Görnitz, N., 03 2017. Ml2motif–reliable extraction of discriminative sequence motifs from learning machines. PLOS ONE 12 (3), 1–22.
[55] Zeiler, M. D., Fergus, R., 2014. Visualizing and understanding convolutional networks. In: Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I. pp. 818–833.
[56] Zhang, J., Lin, Z. L., Brandt, J., Shen, X., Sclaroff, S., 2016. Top-down neural attention by excitation backprop. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV. pp. 543–559.