Digital Signal Processing 73 (2018) 1–15

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

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

ARTICLE INFO

Article history:
Available online 24 October 2017

Keywords:
Deep neural networks
Activation maximization
Sensitivity analysis
Taylor decomposition
Layer-wise relevance propagation

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. As a tutorial paper, the set of methods covered here is not exhaustive, but sufficiently representative to discuss a number of questions in interpretability, technical challenges, and possible applications. The second part of the tutorial focuses on the recently proposed layer-wise relevance propagation (LRP) technique, for which we provide theory, recommendations, and tricks, to make most efficient use of it on real data.

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 [39,62,35]. Techniques for interpreting and understanding what the model has learned have therefore become a key ingredient of a robust validation procedure [68,22,5]. 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 [17,16].

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 [60,74,5,51,55,59].

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 [29,1,65,58].

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. A second part of this tutorial will look in more depth at the recently proposed layer-wise relevance propagation (LRP) technique [5]. 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.

In spite of the practical successes, one should keep in mind that interpreting deep networks remains a young and emerging field of research. There are currently numerous coexisting approaches to interpretability. This tutorial gives a snapshot of the field at present time and it is naturally somewhat biased towards the authors view; as such we hope that it provides useful information to the reader.

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 [43] for a discussion. As a first step, it can be useful to clarify the

* Corresponding authors.
** Corresponding author at: Department of Electrical Engineering & Computer Science, Technische Universität Berlin, Marchstr. 23, Berlin 10587, Germany.
E-mail addresses: gregoire.montavon@tu-berlin.de (G. Montavon), wojciech.samek@hhi.fraunhofer.de (W. Samek), klaus-robert.mueller@tu-berlin.de (K.-R. Müller).

https://doi.org/10.1016/j.dsp.2017.10.011

meaning we assign 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) [5,55]. Post-hoc interpretability should be contrasted to incorporating interpretability directly into the structure of the model, as done, for example, in [54,17,76,37,73].

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. Furthermore, while some methods aim at reaching a comprehensive functional understanding of the model [32,41,8], we will focus here on interpreting the outputs of a DNN, and explaining individual predictions.

Throughout this tutorial, we will 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 [45]), 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).

The features that form the explanation can be supplemented by relevance scores indicating to what extent each feature contributes. Practically, the explanation will be a real-valued vector of same size as the input, where relevant features are given positive scores, and irrelevant features are set to zero.

An explanation can be, for example, a heatmap highlighting which pixels of the input image most strongly support the classification decision [60,34,5]. In natural language processing, explanations can take the form of highlighted text [42,3].

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 [56]. An example of a neural network is shown in Fig. 1.

The 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.

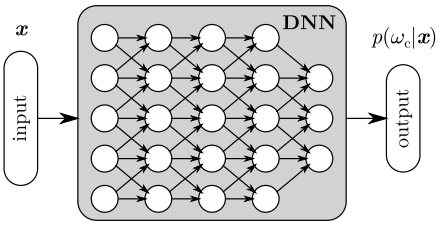

Fig. 1. Example of a neural network composed of many interconnected neurons, and that assigns to the input $\boldsymbol{x}$ a probability of being associated to a certain concept ω_c.

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 [11,19,60].

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 ω_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 [13]. This allows for optimizing the objective by gradient ascent. The rightmost term of the objective is an ℓ_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 [60]. These prototypes, although producing strong class response, can look unnatural.

3.2. Improving AM with an expert

In order to obtain more meaningful prototypes, the ℓ_2-regularizer can be replaced by a more sophisticated one [44,52] called "expert". The expert can be, for example, a model $p(\boldsymbol{x})$ of the data. This leads to the new optimization problem:

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

The prototype $\boldsymbol{x}^\star$ obtained by solving this optimization problem will simultaneously produce strong class response and 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 $\log p(\boldsymbol{x}|\omega_c)$. The learned prototype will thus correspond to the most likely input $\boldsymbol{x}$ for class ω_c.

A possible choice for the expert is the Gaussian RBM [25]. It can represent complex distributions and has a gradient in the input domain. Its log-probability function can be written as:

$$\log p(\boldsymbol{x}) = \textstyle\sum_j f_j(\boldsymbol{x}) - \lambda\|\boldsymbol{x}\|^2 + \text{cst.},$$

where the terms $f_j(\boldsymbol{x}) = \log(1+\exp(\boldsymbol{w}_j^\top \boldsymbol{x} + b_j))$ are learned from the data, and come in superposition to the original ℓ_2-norm regularizer. When interpreting concepts such as natural images classes, more complex density models such as convolutional RBM/DBMs [38], or pixel-RNNs [69] can be used instead. In practice, the choice of the expert $p(\boldsymbol{x})$ plays an important role in determining the appearance of the resulting prototype. The dependence of the prototype on the choice of expert is illustrated in Fig. 2.

On one extreme a coarse expert (**a**) reduces the optimization problem to the maximization of the class probability function $p(\omega_c|\boldsymbol{x})$. On the other extreme an overfitted expert (**d**) essentially reduces the optimization problem to the maximization of the expert $p(\boldsymbol{x})$ itself.

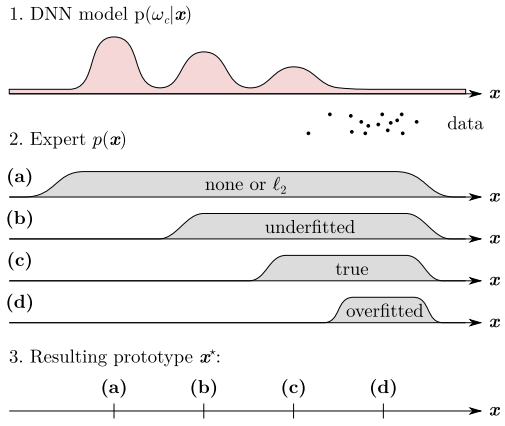

Fig. 2. Cartoon illustrating how the choice of expert $p(\boldsymbol{x})$ affects the prototype $\boldsymbol{x}^\star$ found by AM. The horizontal axis represents the input domain.

When using AM for the purpose of validating a DNN model, an overfitted expert (**d**) must be especially avoided, as the latter could hide interesting failure modes of the DNN model. A slightly underfitted expert (**b**), e.g. that simply favors images with natural colors, can therefore be sufficient. On the other hand, when using AM to gain knowledge on a concept ω_c correctly predicted by the DNN, the focus should be to prevent underfitting. Indeed, an underfitted expert (**b**) 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 ω_c. Hence, it is important in that case to learn a density model as close as possible to the true data distribution (**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 [21]. 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. [51] proposed to build a prototype for ω_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 ℓ_2-norm regularizer in the code space. Once a solution $\boldsymbol{z}^\star$ to the optimization problem is found, the prototype for ω_c is obtained by decoding the solution, that is, $\boldsymbol{x}^\star = g(\boldsymbol{z}^\star)$.

When the code distribution $q(\boldsymbol{z})$ is chosen to be a normal distribution, the ℓ_2 penalty $-\lambda\|\boldsymbol{z}\|^2$ becomes equivalent (up to a scaling factor and a constant) to $\log q(\boldsymbol{z})$, and can therefore be understood as favoring codes with high probability. However, as high probability codes do not necessarily map to high density regions in the input space, the maximization in code space described in this section will only approximately optimize the desired quantity $\log p(\boldsymbol{x}|\omega_c)$.

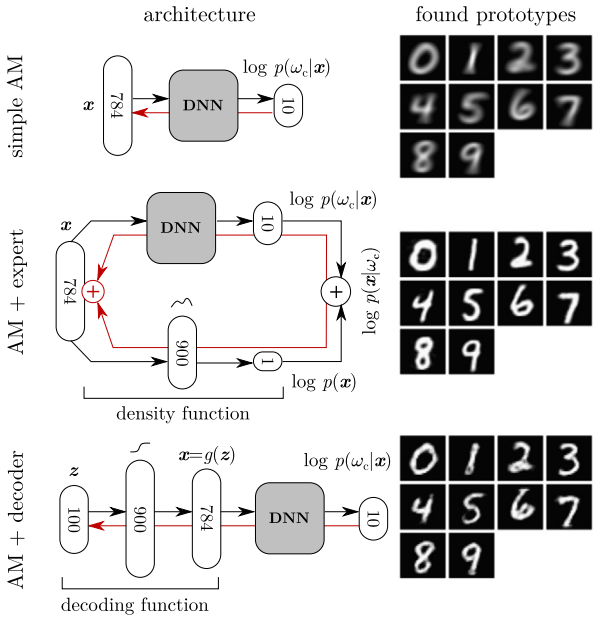

Fig. 3. Architectures supporting AM procedures and found prototypes. Black arrows indicate the forward path and red arrows indicate the reverse path for gradient computation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 ℓ_2-norm regularizer $\lambda\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|^2$ where $\bar{\boldsymbol{x}}$ denotes the data mean for ω_c, (2) a Gaussian RBM expert $p(\boldsymbol{x})$, and (3) a generative model with a two-layer decoding function, and the ℓ_2-norm regularizer $\lambda\|\boldsymbol{z} - \bar{\boldsymbol{z}}\|^2$ where $\bar{\boldsymbol{z}}$ denotes the code mean for ω_c. Corresponding architectures and found prototypes are shown in Fig. 3. 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.

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 ω_c. The issue of multimodality is raised by Nguyen et al. [53], who demonstrate in the context of image classification, the benefit of interpreting a class ω_c using multiple local prototypes instead of a single global one.

Producing an exhaustive description of the modeled concept ω_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 would be 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 η controls the amount of localization. As this parameter increases, the question "*what is a good prototype of* ω_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 ω_c locally, a more relevant question to ask is "*what features of* $\boldsymbol{x}$ *make it representative*

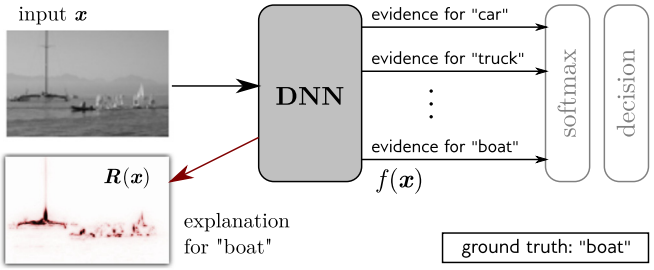

Fig. 4. Explanation of the prediction $f(\boldsymbol{x})$ produced by the DNN for a given data point $\boldsymbol{x}$. Here, $f(\boldsymbol{x})$ represents the evidence for the target class ("boat") as represented by the corresponding neuron just before the softmax layer.

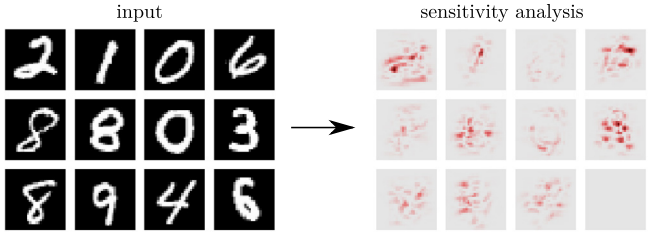

Fig. 5. Sensitivity analysis applied to a convolutional DNN trained on MNIST, and resulting explanations (heatmaps) for selected digits. Red color indicates positive relevance scores. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the concept ω_c?". This question gives rise to a second type of analysis, explaining DNN decisions, 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 ω_c encoded at the output of the deep neural network (DNN). The output neuron that encodes this concept can be described as a function $f(\boldsymbol{x})$ of the input.

A common approach to explanation 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 Fig. 4.

In this example, an image is presented to the DNN, that finds some evidence for class "boat". The prediction 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. In this example, the explanation procedure rightfully assigns relevance to the pixels representing actual boats in the image, and ignores most of the pixels in the background. In the next sections, we present several candidate methods for producing these relevance scores.

4.1. Sensitivity analysis

A first approach to identify the most important input features is sensitivity analysis [77,66,29]. 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

$$R_i(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_i}\right)^2, \tag{1}$$

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 [13,56]. Sensitivity analysis has been regularly used in scientific applications of machine learning such as medical diagnosis [29], ecological modeling [20], or mutagenicity prediction [6]. More recently, it was also used for explaining the classification of images by deep neural networks [60].

Examples of explanations produced by sensitivity analysis when applied to the decisions of a convolutional DNN on MNIST are given in Fig. 5. For each digit $\boldsymbol{x}$, the function $f(\boldsymbol{x})$ to analyze is chosen to be the evidence for the true class.

We can observe that heatmaps are spatially discontinuous and scattered, and do not focus on the actual class-relevant features. This inadequate behavior can be attributed to the nature of sensitivity analysis, which relates to the local variation of the function rather than to the function value itself. More specifically, it is easy to show that relevance scores computed in Equation (1) are a decomposition of the gradient square norm:

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

Thus, sensitivity analysis does not provide an explanation of the function value $f(\boldsymbol{x})$, but of its local slope. In the example above, the heatmap indicates what pixels make the digit belong *more/less* to the target class rather than what makes the digit belong to that class.

4.2. Simple Taylor decomposition

The Taylor decomposition [9,5] 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 $f(\widetilde{\boldsymbol{x}}) = 0$. The root point should remove the information in the input that causes $f(\boldsymbol{x})$ to be positive, e.g. the pattern in a given input image that is responsible for class membership as modeled by the function. Taylor expansion lets us rewrite the function as:

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

where the relevance scores

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

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. A simple two-dimensional function of that class is depicted in Fig. 6 (left). It is composed of linear regions, each of them extending to the origin, and separated by hinges shown as dashed lines. The gradient can be computed everywhere, except on the hinges.

For these functions, we can always find a root point near the origin, $\widetilde{\boldsymbol{x}} = \lim_{\varepsilon \to 0} \varepsilon \cdot \boldsymbol{x}$, 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.

Injecting this root point in the Taylor expansion, the function can be rewritten as

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

where the relevance scores simplify to

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

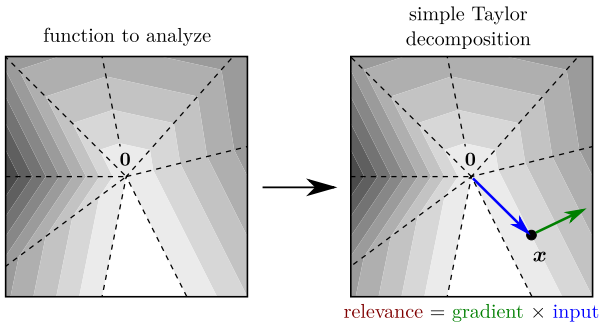

Fig. 6. Function to analyze. High values of the function are shown in dark gray, low values are in light gray, and zero is in white. Gradient and input vectors, on which the analysis relies are shown as green and blue arrows. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

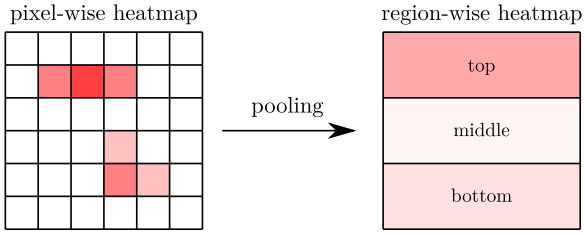

Fig. 7. Illustration of the pooling functionality: The original pixel-wise heatmap is shown on the left. The relevance scores are then pooled in three coarse regions of the image.

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 positively reacts to it, as illustrated in Fig. 6 (right).

The ability of an explanation method to produce a decomposition of $f(\boldsymbol{x})$, offers an interesting additional practical functionality: If the number of input variables is too large for the explanation to be interpretable, the decomposition can be coarsened by *pooling* relevance scores over groups of features $\mathcal{I}$:

$$R_{\mathcal{I}}(\boldsymbol{x}) = \sum_{i\in\mathcal{I}} R_i(\boldsymbol{x}).$$

When the groups of features form a partition of the input features, these pooled relevance scores are becoming themselves a decomposition of the function $f(\boldsymbol{x})$:

$$f(\boldsymbol{x}) = \sum_{\mathcal{I}} R_{\mathcal{I}}(\boldsymbol{x}).$$

Fig. 7 illustrates how from a pixel-wise decomposition, one can produce a coarser decomposition in terms of e.g. spatially meaningful regions.

Lapuschkin et al. [35] used this technique in the context of an image classification task, to determine in what proportion the trained DNN model uses (1) the actual object and (2) its context, to produce a decision.

While we have demonstrated the usefulness of producing a decomposition of $f(\boldsymbol{x})$, one still needs to test whether Taylor decomposition as defined above assigns relevance to the correct input features. Examples of explanations produced by simple Taylor decomposition on a MNIST convolutional DNN are given in Fig. 8.

It can be observed that heatmaps are more complete than those obtained by sensitivity analysis. However, they are characterized by a large amount of negative relevance, here 38% of the total relevance. This is much above the proportion of pixels that truly go against class evidence. This undesirable effect can be attributed to the choice of root point $\widetilde{\boldsymbol{x}} \approx 0$, which is too dissimilar from the actual data point $\boldsymbol{x}$ and thus does not sufficiently contextualize the

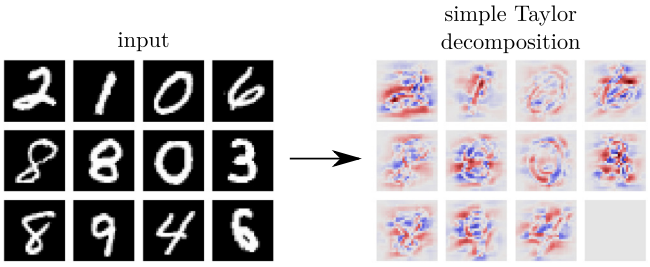

Fig. 8. Simple Taylor decomposition applied to a convolutional DNN trained on MNIST, and resulting explanations. Red and blue colors indicate positive and negative relevance scores. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

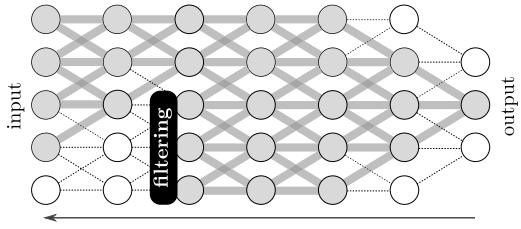

Fig. 9. Graphical depiction of the filtering mechanism. Here, only what passes through the first two neurons of the second hidden layer is further propagated.

explanation. A nearer root point is generally preferable, but is not guaranteed to exist. For example, in the toy example of Fig. 6, only two out of seven linear regions have such nearer root point.

4.3. Backward propagation techniques

An alternative approach to explaining the prediction of a DNN is to make explicit use of its graph structure, and proceed as follows: We start at the output of the network. Then, we move in the graph in reverse direction, progressively mapping the prediction onto the lower layers. The procedure stops once the input of the network is reached. Layer-wise mappings can be engineered for specific properties.

Layer-wise relevance propagation (LRP) [5], for example, is applicable to general network structures including DNNs and kernels. The layer-wise mappings are designed to ensure a relevance conservation property, where the share of $f(\boldsymbol{x})$ received by each neuron is redistributed in same amount on its predecessors. The injection of negative relevance is controlled by hyperparameters. The method is presented in details in Section 5 and is the focus of the second part of this tutorial. For an earlier conserving propagation technique in the context of hierarchical networks, see [34].

Non-conserving backward propagation techniques include deconvolution [74], and its extension guided backprop [63]. Both have been proposed for visualizing the predictions of convolutional DNNs. The first method relies on max-pooling layers to orient the propagated signal on the appropriate locations in the image. The second method relies on the ReLU activations for that purpose. Unlike LRP and other conserving techniques, the visualization produced by these methods cannot directly be interpreted as relevance scores. For comparison purposes, we can however take their absolute values and use them as relevance scores [57].

In comparison to the simple gradient-based methods of Sections 4.1 and 4.2, backward propagation techniques such as LRP or deconvolution were shown empirically to scale better to complex DNN models [57]. These techniques provide a further practical advantage: The quantity being propagated can be *filtered* to only retain what passes through certain neurons or feature maps. The filtering functionality is depicted in Fig. 9.

Table 1
Properties of various techniques for explaining DNN predictions. (⋆) pools variations of f. (†) technically applicable, but no clear interpretation of the result.

	pooling	filtering
sensitivity analysis [60]	✓ (⋆)	
simple Taylor decomposition [5]	✓	
deconvolution [74]	(†)	✓
guided backprop [63]	(†)	✓
layer-wise relevance propagation [5]	✓	✓

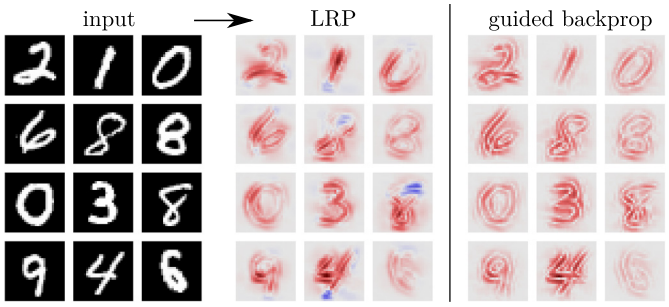

Fig. 10. LRP applied to a convolutional DNN trained on MNIST, and resulting explanations for selected digits. Red and blue colors indicate positive and negative relevance scores respectively. Heatmaps are shown next to those produced by guided backprop. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Filtering can be useful, for example, in the context of multi-task learning, where some hidden neurons are specific to a given task and other neurons are shared. Filtering one or the other set of neurons allows for forming separate explanations for (1) what is specifically relevant to the given task, and (2) what is commonly relevant to all tasks. More generally, filtering can be used to capture multiple components of an explanation, that would otherwise be entangled. A systematic comparison of methods for explaining DNNs is given in Table 1.

5. Layer-wise relevance propagation (LRP)

LRP [5] is a backward propagation technique, specifically designed for explanation. LRP was found to be broadly applicable [35,3,65,2], and to have excellent benchmark performance [57]. The LRP technique is rooted in a conservation principle, where each neuron receives a share of the network output, and redistributes it to its predecessors in equal amount, until the input variables are reached [5,34]. LRP is furthermore embeddable in the theoretical framework of deep Taylor decomposition [46].

Example of explanations produced by LRP are given in Fig. 10. Most of the digit contours are identified as relevant, and a few pixels such as the broken upper-loop of the last digit "8" are identified as negatively relevant. In comparison, non-conserving methods such as guided backprop cannot identify these negatively relevant areas. LRP heatmaps are also easier to interpret than those obtained with the gradient-based methods of Sections 4.1 and 4.2.

A high-level graphical depiction of the LRP procedure applied to a deep neural network is given in Fig. 11.

In the first phase, a standard forward pass is applied to the network and the activations at each layer are collected. In the second phase, the score obtained at the output of the network is propagated backwards in the network, using a set of propagation rules that we provide in Section 5.1. In this layered graph structure, the relevance conservation property can be formulated as follows: Let j and k be indices for neurons of two successive layers. Let R_k be the relevance of neuron k for the prediction $f(\boldsymbol{x})$. We define $R_{j\leftarrow k}$ as the share of R_k that is redistributed to neuron j in the lower layer. The conservation property for this neuron imposes

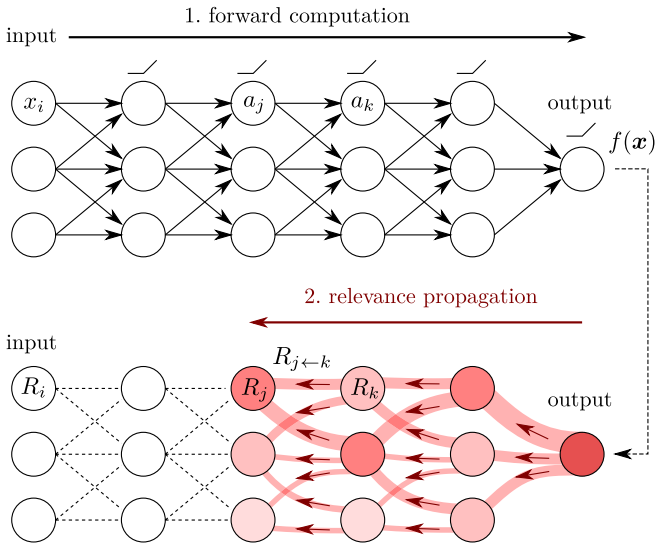

Fig. 11. Diagram of the LRP procedure (here after three steps of redistribution). Red arrows indicate the relevance propagation flow. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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

Likewise, neurons in the lower layer aggregate all relevance coming from the neurons from the higher layer:

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

These two equations, when combined, also ensure a relevance conservation property between layers (proof: $\sum_j R_j = \sum_j \sum_k R_{j\leftarrow k} = \sum_k \sum_j R_{j\leftarrow k} = \sum_k R_k$). Conservation of relevance in the redistribution process also holds globally, so that we can write the chain of equalities

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

where the leftmost and rightmost terms highlight the fact that the method computes a decomposition of $f(\boldsymbol{x})$ in terms of input variables.

5.1. LRP propagation rules

Technically, this conservation property of LRP must be implemented by a specific set of propagation rules. 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 σ is a positive and monotonically increasing activation function.

One propagation rule that is locally conservative and that was shown to work well in practice is the $\alpha\beta$-rule [5] given by:

$$R_j = \sum_k \Big(\alpha \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} - \beta \frac{a_j w_{jk}^-}{\sum_j a_j w_{jk}^-}\Big) R_k, \tag{3}$$

where each term of the sum corresponds to a relevance message $R_{j\leftarrow k}$, where $()^+$ and $()^-$ denote the positive and negative parts respectively, and where the parameters α and β 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

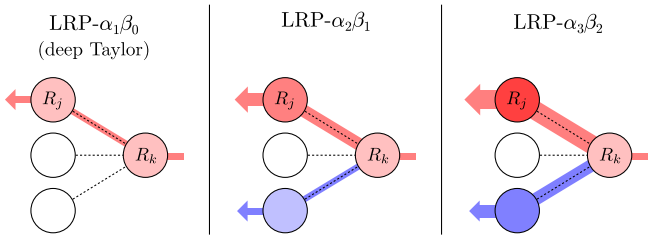

Fig. 12. Graphical depiction of the relevance redistribution process for one neuron, with different parameters α and β. Positive relevance is shown in red. Negative relevance is shown in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

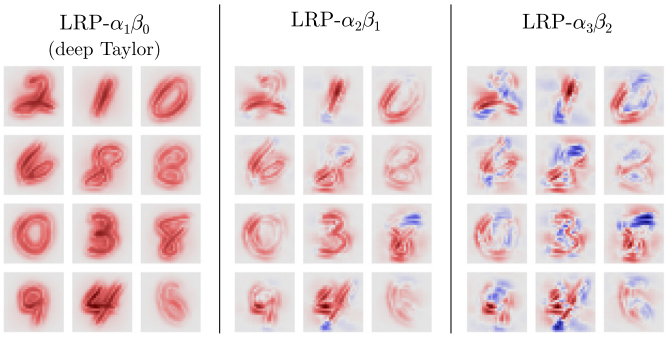

Fig. 13. LRP explanations when choosing different LRP parameters α and β. Positive and negative relevance are shown in red and blue respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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

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 α, β 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.

Fig. 12 depicts the redistribution process for a neuron with positive inputs and weights $(w_{jk})_j = (1, 0, -1)$. The higher α and β, the more positive and negative relevance are being created in the propagation phase.

Examples of explanations obtained with different values of α and β are given in Fig. 13 for MNIST digits predicted by a convolutional DNN. Unless stated otherwise, we use in all experiments the same parameters α and β for each hidden layer, except for the first layer, where we use a pixel-specific rule given later in Eq. (8).

When $\alpha = 1$, the heatmaps contain only positive relevance, and the latter is spread along the contour of the digits in a fairly uniform manner. When choosing $\alpha = 2$, some regions in the images become negatively relevant (e.g. the broken upper-loop of the last digit "8"), but the negative relevance still amounts to only 5% of the total relevance. When setting the higher value $\alpha = 3$, negative relevance starts to appear in a seemingly random fashion, with the share of total relevance surging to 30%. For this simple example, a good choice of propagation rule is LRP-$\alpha_2\beta_1$.

On the deeper BVLC CaffeNet [28] for image recognition, LRP-$\alpha_2\beta_1$ was also shown to work well [5]. For the very deep GoogleNet [67], however, LRP-$\alpha_1\beta_0$ was found to be more stable [46].

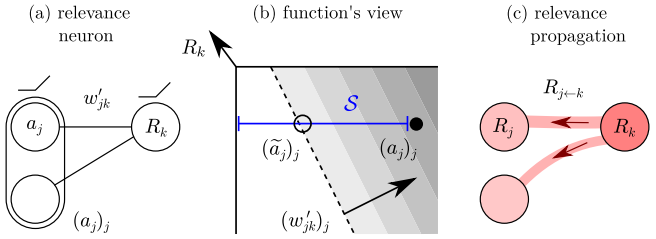

Fig. 14. Diagram of the relevance neuron, its analysis, and the relevance propagation resulting from the analysis. The root search segment is shown in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

When choosing the parameters $\alpha = 1$ and $\beta = 0$, the propagation rule reduces to the simpler rule:

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k. \tag{4}$$

This simpler rule allows for an interpretation of LRP as a deep Taylor decomposition [46], that we present below. The same simple rule was also used later by Zhang et al. [75] 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

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

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. The analysis relies on a special structure of the relevance scores R_k at each layer, which have to be the product of the corresponding neuron activations and positive constant terms. This assumption is necessary in order to apply the Taylor decomposition framework to these neurons. Similarly, the Taylor decomposition procedure must also ensure that resulting relevances in the lower layer have the same product structure, so that relevance can be further propagated backwards.

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"

$$R_k = \max\left(0, \textstyle\sum_j a_j w'_{jk} + b'_k\right), \tag{5}$$

with parameters $w'_{jk} = w_{jk} c_k$ and $b'_k = b_k c_k$. The relevance neuron R_k is shown graphically in Fig. 14(a), and as a function of $(a_j)_j$ in Fig. 14(b). The gradient of grays depicts the neuron's linear activation domain, and the dashed line indicates the hinge of that function.

We now would like to propagate the relevance to the lower layer. For this, we perform a Taylor decomposition of R_k. Because the relevance neuron is linear on its activated domain, one can always take a root point at the limit of zero activation. Thus, the Taylor expansion at this root point contains only first-order terms and is given by

$$R_k = \sum_j \underbrace{\frac{\partial R_k}{\partial a_j}\Big|_{(\widetilde{a}_j)_j} \cdot (a_j - \widetilde{a}_j)}_{R_{j\leftarrow k}}. \tag{6}$$

We search for the nearest root point $(\widetilde{a}_j)_j$ of R_k on the segment $\mathcal{S} = [(a_j 1_{w'_{jk} \leq 0})_j, (a_j)_j]$. The search segment is visualized as a blue line in Fig. 14(b). This search strategy can be interpreted as slowly removing the excitatory inputs until the relevance neuron becomes deactivated. Injecting the found root point $(\widetilde{a}_j)_j$ in Equation (6), one gets the following closed-form expression:

$$R_{j \leftarrow k} = \frac{a_j w^+_{jk}}{\sum_j a_j w^+_{jk}} R_k. \tag{7}$$

The resulting propagation of R_k on the input neurons is illustrated in Fig. 14(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 (4).

We now would like to verify that this relevance redistribution procedure can be repeated one layer below. For this, we inspect the structure of R_j given in Equation (4), 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 imply that similar relevance neuron to the one of Equation (5) 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 [46].

5.3. Handling special layers

In the section above, we have presented the general deep Taylor decomposition principle underlying the propagation of relevance in a sequence of ReLU layers. To ensure a broader applicability for LRP, other layer types need to be considered:

Input layer. The input layer is special in the sense that its input domain differs from the hidden layers. Deep Taylor decomposition adapts to the new input domain by modifying the root search direction $\mathcal{S}$ to remain inside of it [46]. For example, when the input domain is defined by the box constraints $l_i \leq x_i \leq h_i$ (e.g. pixel values restricted between black and white), an appropriate search direction $\mathcal{S}$ will point to the corner of the box opposite to the direction of the weight vector. This results in the new propagation rule:

$$R_i = \sum_j \frac{x_i w_{ij} - l_i w^+_{ij} - h_i w^-_{ij}}{\sum_i x_i w_{ij} - l_i w^+_{ij} - h_i w^-_{ij}} R_j. \tag{8}$$

For a derivation and further propagation rules, see [46]. More complex rules that not only take into consideration the input domain but also the statistics of the data within that domain were recently proposed [30].

Pooling layer. In order to provide spatial or temporal invariance, neural networks often incorporate pooling layers. The ℓ_p-pooling layer has its output defined as

$$a_k = \sqrt[p]{\sum_j a_j^p},$$

where $(a_j)_j$ are the activations inside the pool. A possible propagation rule for this layer is

$$R_{j \leftarrow k} = \frac{a_j}{\sum_j a_j} R_k, \tag{9}$$

i.e. redistribution in proportion to the neuron activations. This rule selects the most relevant neurons in the pool, while also ensuring a continuous transition in the space of pooled activations. This rule also has a deep Taylor interpretation for the case $p = 1$: One first needs to observe that the pooling operation over positive activations is equivalent to a simple ReLU neuron with weights $w_{jk} = 1$, and then observe that Eq. (9) coincides with Eq. (7) for that particular choice of weights.

Other layers. In practical applications, a variety of layers and architectures are used to handle specific types of signals or to reach maximum performance. Examples are LSTMs [26], normalization layers [27,33], or improved element-wise nonlinearities [31]. Although some propagation rules have been shown to work well in practice [5,12,4], it is still an open question whether the deep Taylor decomposition framework can also be extended to these layers.

6. Recommendations and tricks

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 [10,25,47]. Likewise, techniques of interpretation often come with their own set of recommendations and tricks. While this section is mainly focused on LRP, part of the discussion also applies to interpretation techniques in general.

6.1. Structuring the DNN for maximum explainability

We consider here a standard and generally successful class of architectures: DNNs with convolution layers and ReLU neurons. We then ask what specific architectures within that class, can be expected to work optimally with explanation techniques such as LRP. Our first recommendation relates to the global layer-wise structure of the network:

> *Use as few fully-connected layers as needed to be accurate, and train these layers with dropout.*

A reduced number of fully-connected layers avoids that the relevance, when redistributed backwards, looses its connection to the concept being predicted. Training these layers with dropout [64] helps to better align the filters with the actual features, and further encourages the relevance to be redistributed only to the limited set of task-relevant neurons. The second recommendation focuses on spatial or temporal pooling:

> *Use sum-pooling layers abundantly, and prefer them to other types of pooling layers.*

As discussed in Section 5.3, sum-pooling layers are directly embeddable in the deep Taylor decomposition framework. Sum-pooling layers also admit a unique root point in the space of positive activations [46], which allows for an unambiguous choice of LRP redistribution rule Eq. (9). A global sum-pooling layer at the top of a DNN was further advocated by Zhou et al. [76] as a way of spatially localizing class-relevant information. A third recommendation concerns the linear layers in the DNN architecture:

In the linear layers (convolution and fully-connected), constrain biases to be zero or negative.

The use of negative biases strengthens the interpretation of LRP as a deep Taylor decomposition. A second more intuitive argument in favor of negative biases is that they further sparsify the network activations, and in turn, encourage relevance to be redistributed on a limited number of neurons.

6.2. Choosing the LRP rules for explanation

In presence of a deep neural network that follows the recommendations above, or a more general successfully trained DNN, a first set of propagation rules to be tried are the ones derived from deep Taylor decomposition Eqs. (4), (8), (9). These rules exhibit a stable behavior and are also well-understood theoretically.

As a default choice, use the deep Taylor LRP rules.

Eq. (4) in particular, corresponds to the LRP-$\alpha_1\beta_0$ rule and applies to the hidden layers. 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 hyperparameters α, β to more appropriate values.

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 [46], but there is a potentially much larger set of rules that we can choose from. For example, the "ϵ-rule" [5] was applied successfully to text categorization [3,4]. To choose the most appropriate rule among the set of possible ones, one can define a heatmap quality criterion, and select the LRP rules accordingly.

If the heatmaps are still unsatisfactory (or if it is unclear whether they are), consider a larger set of propagation rules, and use the techniques of Section 7 to select the best one.

This places all the weight on the heatmap quality criterion, but can lead in principle to better choices of hyperparameters, potentially different for each layer.

6.3. Tricks for implementing LRP

Although the LRP rules are expressed in this paper for individual neurons, they can be implemented easily and efficiently on large fully-connected or convolution layers. Let us consider, for example, the LRP-$\alpha_1\beta_0$ propagation rule of Equation (4):

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

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	
$z_k \leftarrow \sum_j a_j w^+_{jk}$	$\boldsymbol{z} \leftarrow W_+^\top \cdot \boldsymbol{a}$	(10)
$s_k \leftarrow R_k / z_k$	$\boldsymbol{s} \leftarrow \boldsymbol{R} \oslash \boldsymbol{z}$	(11)
$c_j \leftarrow \sum_k w^+_{jk} s_k$	$\boldsymbol{c} \leftarrow W_+ \cdot \boldsymbol{s}$	(12)
$R_j \leftarrow a_j c_j$	$\boldsymbol{R} \leftarrow \boldsymbol{a} \odot \boldsymbol{c}$	(13)

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 (10) and (12) 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 be 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.

If correctly implemented, and assuming that the convolutions and matrix multiplications are where most of the computation time is spent, the computational complexity of LRP (including forward pass) scales linearly with the forward pass, with some constant factor: $\times 3$ for LRP-$\alpha_1\beta_0$, $\times 5$ for general parameters α and β, and $\times 7$ for the rule of Equation (8) specific to pixel intensities.

Sample code for these propagation rules and for the complete layer-wise propagation procedure is provided at http://heatmapping.org/tutorial. A similar modular approach was also used by Zhang et al. [75] to implement the excitation backprop method. In addition, an LRP toolbox [36] was developed, that can be used to apply LRP to state-of-the-art convolutional DNNs.

6.4. Translation trick for denoising heatmaps

It is sometimes observed that, for classifiers that are not optimally trained or structured, 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 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 τ, τ^{-1} denote the translation and its inverse, and $\mathcal{T}$ is the set of all translations of a few pixels. Note that this translation trick preserves the spatial resolution of the heatmap and is therefore not the same as simple heatmap blurring. This trick was used, for example, by Arbabzadah et al. [2] to reduce the stride artifact of

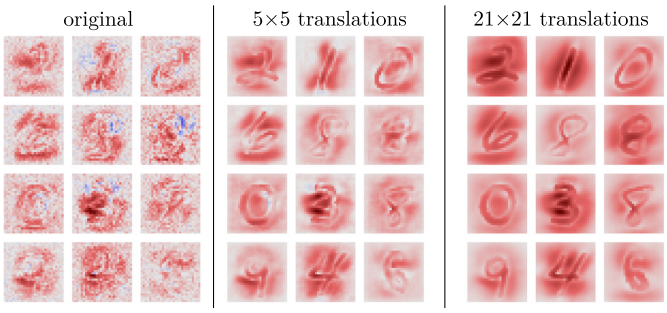

Fig. 15. Heatmaps obtained by explaining a fully-connected DNN on MNIST with LRP-$\alpha_2\beta_1$, and denoised heatmaps resulting from applying translations.

the first convolution layer when explaining facial expression data with LRP.

If choosing too few translations, the denoising effect is limited. If choosing too many translations, the model might receive unexpected inputs, and can consequently produce erratic decisions. Additionally, too much evidence will be assigned to examples for which the prediction is more translation invariant.

The effect of the translation trick on a fully-connected DNN is shown in Fig. 15. The original heatmap is noisy. Adding translations up to two pixels in all directions (5×5 translations) already produces much cleaner heatmaps. Adding more translations further denoises the heatmaps but also changes the amount of evidence attributed to each input image, e.g. the heatmap for the first digit "8" becomes weaker.

The general idea of using translations for denoising is also applicable to other interpretation techniques: For example, the techniques of Section 3 can be enhanced by forcing the class prototype $\boldsymbol{x}^\star$ to produce consistently high responses under small translations. Mordvintsev et al. [50] used a similar trick as part of the inceptionism technique for visualizing DNNs.

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 what the neural network has been trained on and 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 Fig. 16.

When f is a convolutional network, a more direct approach is to add a global top-level pooling layer to the network (after training), and feed the whole image $\boldsymbol{X}$ to it. This direct approach can provide a computational gain compared to the sliding window method. However, it is not strictly equivalent and can produce unreliable heatmaps, e.g. when the network uses border-padded convolutions. In doubt, it is preferable to use the sliding window approach.

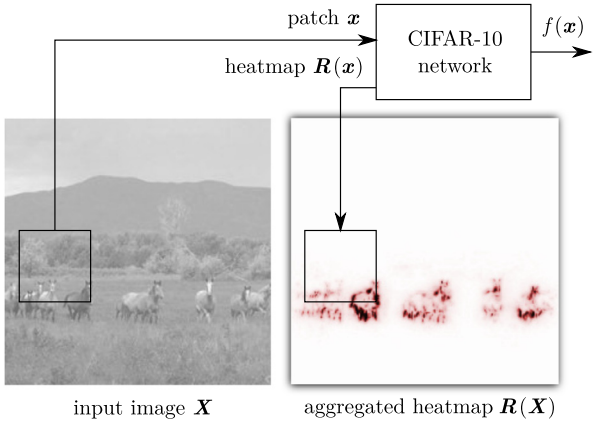

Fig. 16. Highlighting in a large image pixels that are relevant for the CIFAR-10 class "horse", using the sliding window technique.

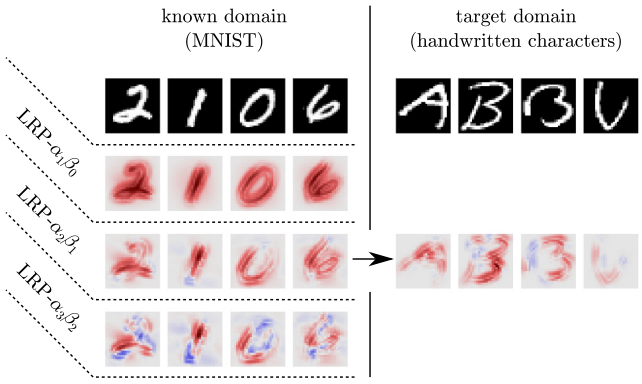

Fig. 17. Transferring the explanation parameters from a known domain (MNIST), to a potentially less known target domain.

7. Evaluating explanation quality

For general tasks, e.g. in the sciences, it can be difficult to determine objectively whether an explanation technique is good or not, as the concept predicted by the DNN may only be interpretable by an expert. Here, we present some strategies to systematically and objectively assess the quality of explanations. Section 7.1 discusses how a simple related task can serve as a proxy for that purpose. Sections 7.2 and 7.3 discuss how to perform such quality assessement by looking analytically at the explanation function and its relation to the prediction.

7.1. Transfer with a simple task

Consider the case where the task of interest is related to a known simple task (e.g. same input domain, same structure of the prediction, and similar neural network architecture needed to solve it). On the simple task, it is usually easier to determine whether an explanation is good or bad, as the task typically involves daily-life concepts for which we know what are the important features. The simple task can therefore be used as a proxy for the task of interest, in order to evaluate explanation quality.

This idea is illustrated in Fig. 17, in the context of selecting the most appropriate parameters of an explanation technique. In this example, we would like to explain the prediction of a DNN trained on a handwritten characters dataset [70]. We first find the parameters that best explain the prediction of a DNN trained on the simple and related MNIST data. Then, the found parameters are applied to the original problem.

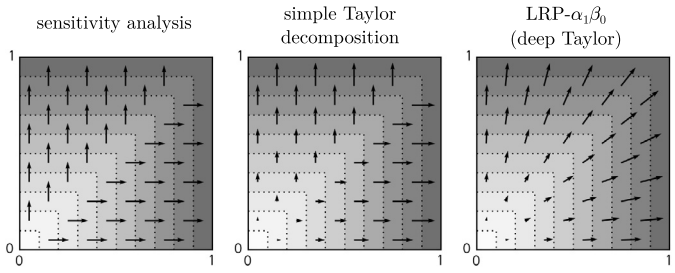

Fig. 18. Explaining $\max(x_1, x_2)$. Function values are represented as a contour plot, with dark regions corresponding to high values. Relevance scores are represented as a vector field, where horizontal and vertical components are the relevance of respective input variables.

We can observe that the heatmaps look similar across domains, with a similar placement of relevance around the handwriting strokes and a similar share of negative relevance. In the target domain, LRP identifies important features such as the upper triangle for the character "A", and the two vertically stacked rounded strokes for the character "B". It also identifies as negatively relevant a defect in the second character "B", where the vertical bar is too distant.

However, when a simple related task is *not* available, it becomes essential to be able 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. We present in the following two important properties of an explanation, along with possible evaluation metrics.

7.2. Explanation continuity

A 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:

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

The problem of explanation continuity is first illustrated in Fig. 18 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 analysis and simple Taylor decomposition are discontinuous on the line $x_1 = x_2$. Here, only LRP produces a smooth transition.

Techniques that rely on the function's gradient, such as sensitivity analysis or simple Taylor decomposition, are also strongly exposed to derivative noise [61] that characterizes complex machine learning models. More specifically, they are subject to the problem of shattered gradients [7,49] occurring in deep ReLU networks: The number of piecewise-linear regions tends to grow very quickly with depth, and thus cause the gradient to become strongly discontinuous and loose its information content.

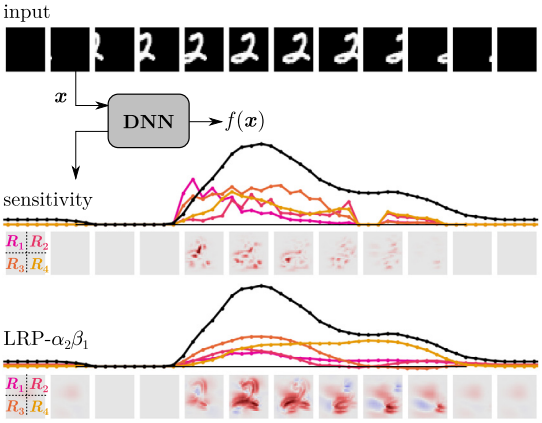

Fig. 19. Classification "2" by a DNN, explained by different methods, as we move a handwritten digit from left to right. The function value is shown in black and relevance scores (pooled in four quadrants) are shown in colors from pink to orange. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 19 looks at the continuity of relevance scores for sensitivity analysis and LRP-$\alpha_2\beta_1$, when applying them to a convolutional DNN trained on MNIST. We move in the input space by slowly translating a MNIST digit from left to right, and keep track of the function value and the relevance scores of the two explanation methods.

Although the function $f(\boldsymbol{x})$ is continuous, relevance scores produced by sensitivity analysis are strongly varying. Here again, only LRP produces continuous explanations.

7.3. Explanation selectivity

Another desirable property of an explanation is that it redistributes relevance to variables that have the strongest impact on the function value $f(\boldsymbol{x})$. Bach et al. [5] and Samek et al. [57] 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" [5,57], 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:

> Sort features from most to least relevant ($R_1 \geq \cdots \geq R_d$)
>
> **for** i from 1 to d:
>
> - record the current function value $f(\boldsymbol{x})$
> - remove the ith most relevant 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, summarized by a low AUC score indicates that the correct features have been identified as relevant. Plots and AUC results can be averaged over a large number of examples in the dataset.

Fig. 20 illustrates the procedure on the same DNN as in Fig. 19. At each iteration, a patch of size 4×4 corresponding to the region with highest relevance is set to black. The plot keeps track of the function value $f(\boldsymbol{x})$ as the features are being progressively removed and computes an average over a large number of examples. The plot indicates that LRP and guided backprop are more selective than sensitivity analysis and simple Taylor decomposition, and

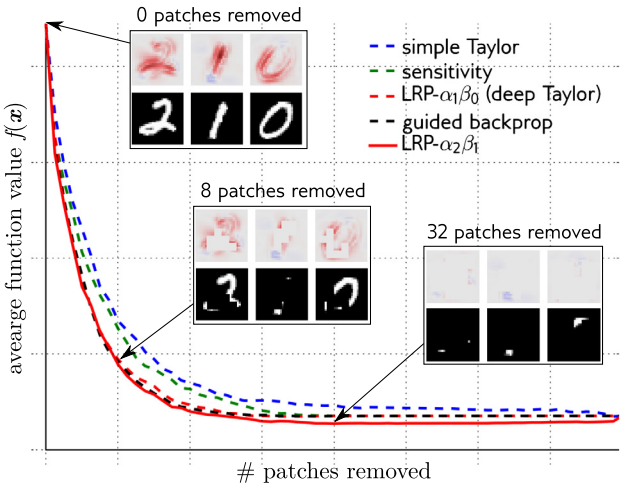

Fig. 20. Illustration of the "pixel-flipping" procedure. At each step, the next most relevant region according to the heatmap is removed (by setting it to black), and the function value $f(\boldsymbol{x})$ is recorded.

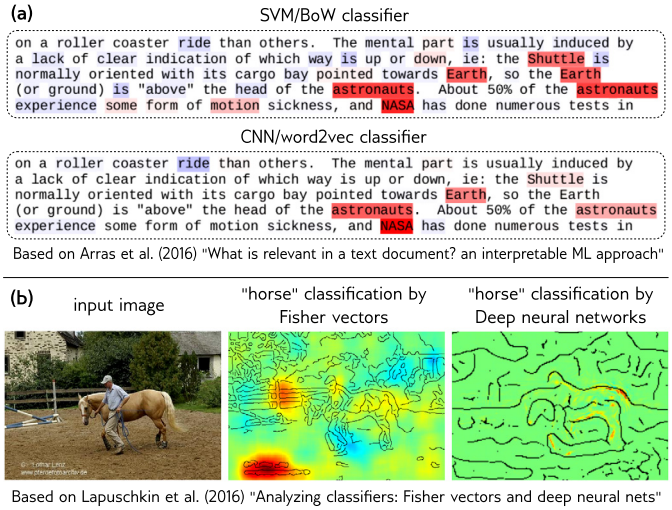

Fig. 21. Examples taken from the literature of model validation via explanation. (a) Explanation of the concept "`sci.space`" by two text classifiers. (b) Unexpected use of copyright tags by the Fisher vector model for predicting the class "horse".

that LRP-$\alpha_2\beta_1$ scores the best among all methods. This success can be attributed to the ability of LRP to model negative evidence. This negative evidence is preserved during the whole pixel-flipping procedure, thus further lowering the function value $f(\boldsymbol{x})$. We refer to Samek et al. [57] for a more comprehensive comparison between different explanation methods.

The choice of a patch instead of a pixel as a unit of removal lets the analysis focus on removing actual content of the image, and avoids introducing pixel artifacts. For natural images, various patch replacement strategies (e.g. gray patch, random noise, Gaussian blur) as well as various patch sizes (up to 19×19 pixels) have been used in practice [57]. For text categorization, Arras et al. [3] choose the word as a unit of feature removal, and remove words by setting their associated word2vec vector to zero.

It is important to note that the result of the analysis depends to some extent on the feature removal process. As mentioned above, various feature removal strategies can be used, but a general rule is that it should keep as much as possible the data point being modified on the data manifold. Indeed, 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 evaluate a machine learning model in practice, the validation error is only a proxy for the true error, as the validation set might differ statistically from the true distribution. A human inspection of the model rendered interpretable can thus be a good complement to the basic 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. A LRP procedure was applied to each model to produce the explanations. The authors 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. Fig. 21(a) displays explanations for an example of the target class `sci.space`.

Lapuschkin et al. [35] 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. A LRP procedure was applied to explain the predictions of both models. Explanations for a given image are shown in Fig. 21(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.

Once a weakness of the model has been identified by say LRP, various countermeasures can be taken to improve the model. For example, the reliance of the model on a data artifact (e.g. a copyright tag) can be mitigated by removing it (or injecting similar artifact in other classes), and retraining the model. A model that decides based on too many input variables can be retrained with a sparsity penalty. The rich feedback provided by explanation allows in principle to explore the space of DNN models in a more guided manner than a validation procedure based only on classification error.

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

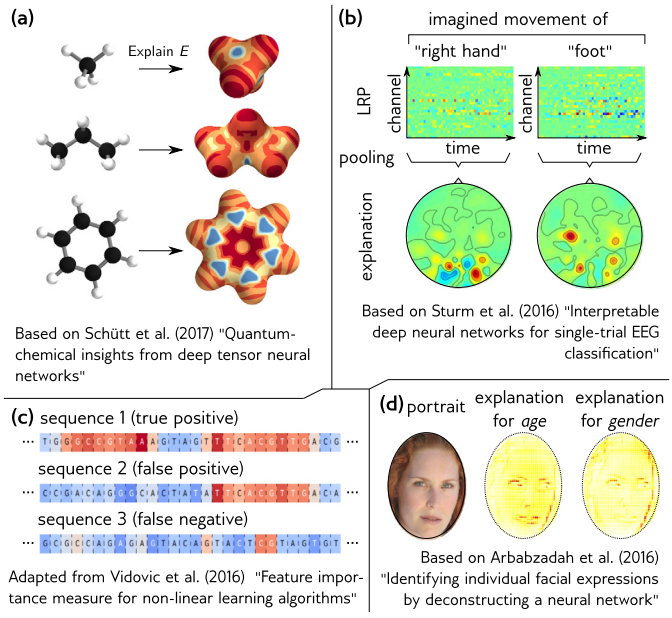

Fig. 22. Overview of several applications of machine learning explanation techniques in the sciences. (a) Molecular response maps for quantum chemistry, (b) EEG heatmaps for neuroimaging, (c) extracting relevant information from gene sequences, (d) analysis of facial appearance.

a tradeoff between interpretability and predictive power, where linear models would sometimes be preferred to nonlinear models despite their typically 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 [48,23,58,18]. These models have been trained in a data-driven manner, without simulated physics involved into the prediction. In particular, Schütt et al. [58] 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 Fig. 22(a) for various molecules.

Sturm et al. [65] 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 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. Examples of these individual EEG brain maps are given in Fig. 22(b). For classical linear explanation of neural activation patterns in cognitive brain science experiments or Brain Computer Interfacing, see [15,40,14,24].

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 [71,72]. They 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 Fig. 22(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 Fig. 22(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* 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 can be applied to a general class of nonlinear machine learning models, no matter how they were trained and who trained them – even for fully trained models that are available for download like BVLC CaffeNet [28] or GoogleNet [67].

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 Alexan-

der 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] B. Alipanahi, A. Delong, M.T. Weirauch, B.J. Frey, Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning, Nat. Biotechnol. 33 (8) (Jul 2015) 831–838.
[2] F. Arbabzadah, G. Montavon, K.-R. Müller, W. Samek, Identifying individual facial expressions by deconstructing a neural network, in: Pattern Recognition - 38th German Conference, GCPR 2016, Hannover, Germany, 12–15 September, 2016, Proceedings, 2016, pp. 344–354.
[3] L. Arras, F. Horn, G. Montavon, K.-R. Müller, W. Samek, "What is relevant in a text document?": an interpretable machine learning approach, PLoS ONE 12 (8) (2017) e0181142.
[4] L. Arras, G. Montavon, K. Müller, W. Samek, Explaining recurrent neural network predictions in sentiment analysis, in: Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EMNLP 2017, Copenhagen, Denmark, 8 September, 2017, 2017, pp. 159–168.
[5] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation, PLoS ONE 10 (7) (2015) e0130140.
[6] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller, How to explain individual classification decisions, J. Mach. Learn. Res. 11 (2010) 1803–1831.
[7] D. Balduzzi, M. Frean, L. Leary, J.P. Lewis, K.W.-D. Ma, B. McWilliams, The shattered gradients problem: if resnets are the answer, then what is the question?, in: D. Precup, Y.W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 70, PMLR, International Convention Centre, Sydney, Australia, Aug 2017, pp. 342–350.
[8] D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network dissection: quantifying interpretability of deep visual representations, CoRR, arXiv:1704.05796, 2017.
[9] S. Bazen, X. Joutard, The Taylor Decomposition: A Unified Generalization of the Oaxaca Method to Nonlinear Models, Working papers, HAL, 2013.
[10] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in: Neural Networks: Tricks of the Trade, second edition, 2012, pp. 437–478.
[11] P. Berkes, L. Wiskott, On the analysis and interpretation of inhomogeneous quadratic forms as receptive fields, Neural Comput. 18 (8) (2006) 1868–1895.
[12] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, W. Samek, 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, 6–9 September, 2016, Proceedings, Part II, 2016, pp. 63–71.
[13] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Inc., New York, NY, USA, 1995.
[14] B. Blankertz, S. Lemm, M.S. Treder, S. Haufe, K.-R. Müller, Single-trial analysis and classification of ERP components — A tutorial, NeuroImage 56 (2) (2011) 814–825.
[15] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, K.-R. Müller, Optimizing spatial filters for robust EEG single-trial analysis, IEEE Signal Process. Mag. 25 (1) (2008) 41–56.
[16] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L.D. Jackel, U. Muller, Explaining how a deep neural network trained with end-to-end learning steers a car, CoRR, arXiv:1704.07911, 2017.
[17] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, N. Elhadad, 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, 10–13 August, 2015, 2015, pp. 1721–1730.
[18] S. Chmiela, A. Tkatchenko, H.E. Sauceda, I. Poltavsky, K.T. Schütt, K.-R. Müller, Machine learning of accurate energy-conserving molecular force fields, Sci. Adv. 3 (5) (May 2017) e1603015.
[19] D. Erhan, Y. Bengio, A. Courville, P. Vincent, Visualizing Higher-Layer Features of a Deep Network, Tech. Rep. 1341, University of Montreal, Jun. 2009, also presented at the ICML 2009 Workshop on Learning Feature Hierarchies, Montréal, Canada.
[20] M. Gevrey, I. Dimopoulos, S. Lek, Review and comparison of methods to study the contribution of variables in artificial neural network models, Ecol. Model. 160 (3) (Feb 2003) 249–264.
[21] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A.C. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, Quebec, Canada, 8–13 December, 2014, 2014, pp. 2672–2680.
[22] K. Hansen, D. Baehrens, T. Schroeter, M. Rupp, K.-R. Müller, Visual interpretation of kernel-based prediction models, Mol. Inform. 30 (9) (Sep 2011) 817–826.
[23] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O.A. von Lilienfeld, K.-R. Müller, A. Tkatchenko, Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space, J. Phys. Chem. Lett. 6 (12) (Jun 2015) 2326–2331.
[24] S. Haufe, F.C. Meinecke, K. Görgen, S. Dähne, J.-D. Haynes, B. Blankertz, F. Bießmann, On the interpretation of weight vectors of linear models in multivariate neuroimaging, NeuroImage 87 (2014) 96–110.
[25] G.E. Hinton, A practical guide to training restricted Boltzmann machines, in: Neural Networks: Tricks of the Trade, second edition, 2012, pp. 599–619.
[26] S. Hochreiter, J. Schmidhuber, LSTM can solve hard long time lag problems, in: Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, 2–5 December, 1996, 1996, pp. 473–479.
[27] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition?, in: IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, 27 September – 4 October, 2009, 2009, pp. 2146–2153.
[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R.B. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the ACM International Conference on Multimedia, MM'14, Orlando, FL, USA, 3–7 November, 2014, 2014, pp. 675–678.
[29] J. Khan, J.S. Wei, M. Ringnér, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C.R. Antonescu, C. Peterson, P.S. Meltzer, Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks, Nat. Med. 7 (6) (Jun 2001) 673–679.
[30] P.-J. Kindermans, K.T. Schütt, M. Alber, K.-R. Müller, S. Dähne, PatternNet and patternLRP – improving the interpretability of neural networks, CoRR, arXiv:1705.05598, 2017.
[31] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, CoRR, arXiv:1706.02515, 2017.
[32] R. Krishnan, G. Sivakumar, P. Bhattacharya, Extracting decision trees from trained neural networks, Pattern Recognit. 32 (12) (1999) 1999–2009.
[33] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information, Processing Systems 2012, Proceedings of a meeting held 3–6 December, 2012, Lake Tahoe, Nevada, United States, 2012, pp. 1106–1114.
[34] W. Landecker, M.D. Thomure, L.M.A. Bettencourt, M. Mitchell, G.T. Kenyon, S.P. Brumby, Interpreting individual classifications of hierarchical networks, in: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16–19 April, 2013, 2013, pp. 32–38.
[35] S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, W. Samek, Analyzing classifiers: Fisher vectors and deep neural networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas NV, USA, 27–30 June, 2016, 2016, pp. 2912–2920.
[36] S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, W. Samek, The layer-wise relevance propagation toolbox for artificial neural networks, J. Mach. Learn. Res. 17 (114) (2016) 1–5.
[37] H. Larochelle, G.E. Hinton, Learning to combine foveal glimpses with a third-order Boltzmann machine, in: Advances in Neural Information Processing Systems, vol. 23, 2010, pp. 1243–1251.
[38] H. Lee, R.B. Grosse, R. Ranganath, A.Y. Ng, 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, 14–18 June, 2009, 2009, pp. 609–616.
[39] J.T. Leek, R.B. Scharpf, H.C. Bravo, D. Simcha, B. Langmead, W.E. Johnson, D. Geman, K. Baggerly, R.A. Irizarry, Tackling the widespread and critical impact of batch effects in high-throughput data, Nat. Rev. Genet. 11 (10) (Sep 2010) 733–739.
[40] S. Lemm, B. Blankertz, T. Dickhaus, K.-R. Müller, Introduction to machine learning for brain imaging, NeuroImage 56 (2) (2011) 387–399.
[41] B. Letham, C. Rudin, T.H. McCormick, D. Madigan, Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model, Ann. Appl. Stat. 9 (3) (Sep 2015) 1350–1371.
[42] J. Li, X. Chen, E.H. Hovy, D. Jurafsky, 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, 12–17 June, 2016, 2016, pp. 681–691.
[43] Z.C. Lipton, The mythos of model interpretability, CoRR, arXiv:1606.03490, 2016.
[44] A. Mahendran, A. Vedaldi, Understanding deep image representations by inverting them, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June, 2015, 2015, pp. 5188–5196.

[45] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, 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 5–8 December, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 3111–3119.
[46] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller, Explaining nonlinear classification decisions with deep Taylor decomposition, Pattern Recognit. 65 (2017) 211–222.
[47] G. Montavon, G. Orr, K.-R. Müller, Neural Networks: Tricks of the Trade, 2nd edition, Springer Publishing Company, Inc., 2012.
[48] G. Montavon, M. Rupp, V. Gobre, A. Vazquez-Mayagoitia, K. Hansen, A. Tkatchenko, Machine learning of molecular electronic properties in chemical compound space, New J. Phys. 15 (9) (Sep 2013) 095003.
[49] G.F. Montúfar, R. Pascanu, K. Cho, Y. Bengio, On the number of linear regions of deep neural networks, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, Quebec, Canada, 8–13 December, 2014, 2014, pp. 2924–2932.
[50] A. Mordvintsev, C. Olah, M. Tyka, Inceptionism: going deeper into neural networks, http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html, Jun. 2015.
[51] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, J. Clune, 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, Barcelona, Spain, 5–10 December, 2016, 2016, pp. 3387–3395.
[52] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, J. Clune, Plug & play generative networks: conditional iterative generation of images in latent space, CoRR, arXiv:1612.00005, 2016.
[53] A. Nguyen, J. Yosinski, J. Clune, Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks, CoRR, arXiv:1602.03616, 2016.
[54] B. Poulin, R. Eisner, D. Szafron, P. Lu, R. Greiner, D.S. Wishart, A. Fyshe, B. Pearcy, C. Macdonell, J. Anvik, 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, Boston, Massachusetts, USA, 16–20 July, 2006, 2006, pp. 1822–1829.
[55] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August, 2016, 2016, pp. 1135–1144.
[56] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (Oct 1986) 533–536.
[57] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, Evaluating the visualization of what a deep neural network has learned, IEEE Trans. Neural Netw. Learn. Syst. 28 (11) (2017) 2660–2673.
[58] K.T. Schütt, F. Arbabzadah, S. Chmiela, K.R. Müller, A. Tkatchenko, Quantum-chemical insights from deep tensor neural networks, Nat. Commun. 8 (Jan 2017) 13890.
[59] R.R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra, Grad-CAM: why did you say that? Visual explanations from deep networks via gradient-based localization, CoRR, arXiv:1610.02391, 2016.
[60] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: visualising image classification models and saliency maps, CoRR, arXiv:1312.6034, 2013.
[61] J.C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, K. Burke, Finding density functionals with machine learning, Phys. Rev. Lett. 108 (25) (Jun 2012).
[62] C. Soneson, S. Gerster, M. Delorenzi, Batch effect confounding leads to strong bias in performance estimates obtained by cross-validation, PLoS ONE 9 (6) (2014) e0100335.
[63] J.T. Springenberg, A. Dosovitskiy, T. Brox, M.A. Riedmiller, Striving for simplicity: the all convolutional net, CoRR, arXiv:1412.6806, 2014.
[64] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[65] I. Sturm, S. Lapuschkin, W. Samek, K.-R. Müller, Interpretable deep neural networks for single-trial EEG classification, J. Neurosci. Methods 274 (Dec 2016) 141–145.
[66] A. Sung, Ranking importance of input parameters of neural networks, Expert Syst. Appl. 15 (1998) 405–411.
[67] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S.E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, 7–12 June, 2015, 2015, pp. 1–9.
[68] B.J. Taylor, Methods and Procedures for the Verification and Validation of Artificial Neural Networks, Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2005.
[69] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, 19–24 June, 2016, 2016, pp. 1747–1756.
[70] L. van der Maaten, A New Benchmark Dataset for Handwritten Character Recognition, Tech. Rep. TiCC TR 2009-002, Tilburg University, 2009.
[71] M.M.-C. Vidovic, N. Görnitz, K.-R. Müller, M. Kloft, Feature importance measure for non-linear learning algorithms, CoRR, arXiv:1611.07567, 2016.
[72] M.M.-C. Vidovic, M. Kloft, K.-R. Müller, N. Görnitz, Ml2motif-reliable extraction of discriminative sequence motifs from learning machines, PLoS ONE 12 (3) (2017) e0174392.
[73] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio, Show, attend and tell: neural image caption generation with visual attention, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 2048–2057.
[74] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision – ECCV 2014 – 13th European Conference, Zurich, Switzerland, 6–12 September, 2014, Proceedings, Part I, 2014, pp. 818–833.
[75] J. Zhang, Z.L. Lin, J. Brandt, X. Shen, S. Sclaroff, Top-down neural attention by excitation backprop, in: Computer Vision – ECCV 2016 – 14th European Conference, Amsterdam, The Netherlands, 11–14 October, 2016, Proceedings, Part IV, 2016, pp. 543–559.
[76] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas NV, USA, 27–30 June, 2016, 2016, pp. 2921–2929.
[77] J.M. Zurada, A. Malinowski, I. Cloete, Sensitivity analysis for minimization of input data dimension for feedforward neural network, in: 1994 IEEE International Symposium on Circuits and Systems, ISCAS 1994, London, England, UK, 30 May – 2 June, 1994, 1994, pp. 447–450.

Grégoire Montavon received a Masters degree in Communication Systems from École Polytechnique Fédérale de Lausanne, in 2009 and a Ph.D. degree in Machine Learning from the Technische Universität Berlin, in 2013. He is currently a Research Associate in the Machine Learning Group at TU Berlin.

Wojciech Samek received a Diploma degree in Computer Science from Humboldt University Berlin in 2010 and the Ph.D. degree in Machine Learning from Technische Universität Berlin, in 2014. Currently, he directs the Machine Learning Group at Fraunhofer Heinrich Hertz Institute. His research interests include neural networks and signal processing.

Klaus-Robert Müller (Ph.D. 92) has been a Professor of computer science at TU Berlin since 2006; co-director Berlin Big Data Center. He won the 1999 Olympus Prize of German Pattern Recognition Society, the 2006 SEL Alcatel Communication Award, and the 2014 Science Prize of Berlin. Since 2012, he is an elected member of the German National Academy of Sciences – Leopoldina.