10
Layer-Wise Relevance Propagation: An Overview

Grégoire Montavon[1], Alexander Binder[2], Sebastian Lapuschkin[3], Wojciech Samek[3], and Klaus-Robert Müller[1,4,5]

[1] Technische Universität Berlin, 10587 Berlin, Germany
{gregoire.montavon,klaus-robert.mueller}@tu-berlin.de
[2] Singapore University of Technology and Design, 487372 Singapore
alexander_binder@sutd.edu.sg
[3] Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany
{sebastian.lapuschkin,wojciech.samek}@hhi.fraunhofer.de
[4] Korea University, Anam-dong, Seongbuk-gu, Seoul 02841, Korea
[5] Max Planck Institute for Informatics, 66123 Saarbrücken, Germany

Abstract. For a machine learning model to generalize well, one needs to ensure that its decisions are supported by meaningful patterns in the input data. A prerequisite is however for the model to be able to explain itself, e.g. by highlighting which input features it uses to support its prediction. Layer-wise Relevance Propagation (LRP) is a technique that brings such explainability and scales to potentially highly complex deep neural networks. It operates by propagating the prediction backward in the neural network, using a set of purposely designed propagation rules. In this chapter, we give a concise introduction to LRP with a discussion of (1) how to implement propagation rules easily and efficiently, (2) how the propagation procedure can be theoretically justified as a 'deep Taylor decomposition', (3) how to choose the propagation rules at each layer to deliver high explanation quality, and (4) how LRP can be extended to handle a variety of machine learning scenarios beyond deep neural networks.

Keywords: Explanations · Deep Neural Networks · Layer-Wise Relevance Propagation · Deep Taylor Decomposition

10.1 Introduction

Machine learning techniques such as deep neural networks have reached many successes in scientific [9, 33, 45, 14, 17] and industrial (e.g. [2, 20, 32]) applications. A main driver for the adoption of these techniques is the rise of large datasets, enabling the extraction of complex real-world correlations and nonlinearities.

Large datasets, however, are often plagued by the presence of spurious correlations between the different variables [13]. Spurious correlations leave the learning machine perplexed when having to decide which of the few correlated

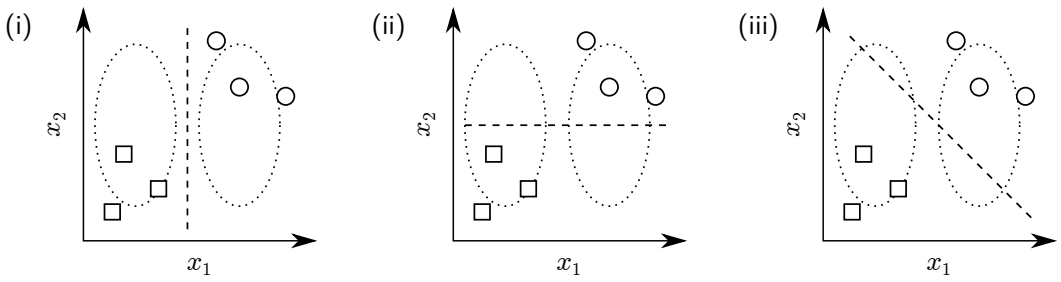

Fig. 10.1. Illustration of the problem of spurious correlations often encountered in high-dimensional data. In this example, both x_1 and x_2 predict the current data, but only x_1 generalizes correctly to the true distribution.

input variables should be used to support the prediction. A simple example is given in Fig. 10.1. The model classifies the data perfectly by using either feature x_1, feature x_2, or both of them, yet only the first option will generalize correctly to new data. Failure to learn the correct input features may lead to 'Clever Hans'-type predictors [30].

Feature selection [19] offers a potential solution by presenting to the learning machine only a limited number of 'good' input features. This approach is however difficult to apply e.g. in image recognition, where the role of individual pixels is not fixed.

Explainable machine learning [52, 8, 37] looks at the problem in the other direction: First, a model is trained without caring too much about feature selection. Only after training we look at which input features the neural network has learned. Based on this explanatory feedback, 'bad' features can be removed and the model can be retrained on the cleaned data [15, 57]. A simple method, Taylor Decomposition [11, 7], produces explanations by performing a Taylor expansion of the prediction $f(\boldsymbol{x})$ at some nearby reference point $\widetilde{\boldsymbol{x}}$:

$$f(\boldsymbol{x}) = f(\widetilde{\boldsymbol{x}}) + \sum_{i=1}^{d} (x_i - \widetilde{x}_i) \cdot [\nabla f(\widetilde{\boldsymbol{x}})]_i + \ldots$$

First-order terms (elements of the sum) quantify the relevance of each input feature to the prediction, and form the explanation. Although simple and straightforward, this method is unstable when applied to deep neural networks. The instability can be traced to various known shortcomings of deep neural network functions:

- *Shattered gradients* [10]: While the function value $f(\boldsymbol{x})$ is generally accurate, the gradient of the function is noisy.
- *Adversarial examples* [53]: Some tiny perturbations of the input $\boldsymbol{x}$ can cause the function value $f(\boldsymbol{x})$ to change drastically.

These shortcomings make it difficult to choose a meaningful reference point $\widetilde{\boldsymbol{x}}$ with a meaningful gradient $\nabla f(\widetilde{\boldsymbol{x}})$. This prevents the construction of a reliable explanation [37].

Numerous explanation techniques have been proposed to better address the complexity of deep neural networks. Some proposals improve the explanation

by integrating a large number of local gradient estimates [49, 51]. Other techniques replace the gradient by a coarser estimate of effect [60], e.g. the model response to patch-like perturbations [58]. Further techniques involve the optimization of some local surrogate model [41], or of the explanation itself [18]. All these techniques involve multiple neural network evaluations, which can be computationally expensive.

In the following, we place our focus on Layer-wise Relevance Propagation [7], a technique that leverages the graph structure of the deep neural network to quickly and reliably compute explanations.

10.2 Layer-Wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) [7] is an explanation technique applicable to models structured as neural networks, where inputs can be e.g. images, videos, or text [7, 3, 5]. LRP operates by propagating the prediction $f(\boldsymbol{x})$ backward in the neural network, by means of purposely designed local propagation rules.

The propagation procedure implemented by LRP is subject to a conservation property, where what has been received by a neuron must be redistributed to the lower layer in equal amount. This behavior is analogous to Kirchoff's conservation laws in electrical circuits, and shared by other works on explanations such as [27, 46, 59]. Let j and k be neurons at two consecutive layers of the neural network. Propagating relevance scores $(R_k)_k$ at a given layer onto neurons of the lower layer is achieved by applying the rule:

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k.$$

The quantity z_{jk} models the extent to which neuron j has contributed to make neuron k relevant. The denominator serves to enforce the conservation property. The propagation procedure terminates once the input features have been reached. If using the rule above for all neurons in the network, it is easy to verify the layer-wise conservation property $\sum_j R_j = \sum_k R_k$, and by extension the global conservation property $\sum_i R_i = f(\boldsymbol{x})$. The overall LRP procedure is illustrated in Fig. 10.2.

Although LRP clearly differs from the simple Taylor decomposition approach mentioned in the introduction, we will observe in Section 10.2.3 that each step of the propagation procedure can be modeled as an own Taylor decomposition performed over local quantities in the graph [36].

LRP was applied to discover biases in commonly used ML models and datasets [28, 30]. It was also applied to extract new insights from well-functioning ML models, e.g. in face expression recognition [4, 29]. LRP was used to find relevant features for audio source localization [39], to identify points of interest in side channel traces [21], and to identify EEG patterns that explain decisions in brain-computer interfaces [50]. In the biomedical domain, LRP was used to

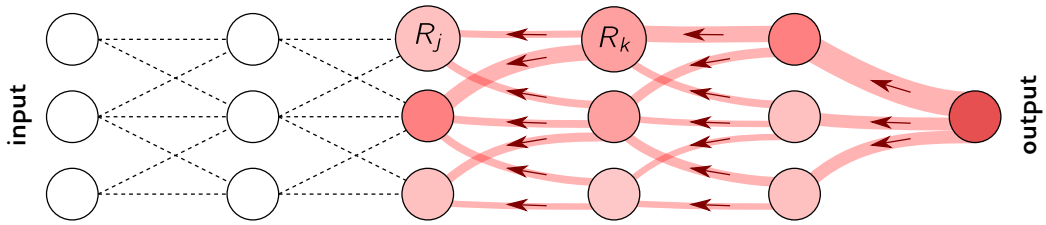

Fig. 10.2. Illustration of the LRP procedure. Each neuron redistributes to the lower layer as much as it has received from the higher layer.

identify subject-specific characteristics in gait patterns [24], to highlight relevant cell structure in microscopy [12], as well as to explain therapy predictions [56]. Finally, an extension called CLRP was applied to highlight relevant molecular sections in the context of protein-ligand scoring [23].

10.2.1 LRP Rules for Deep Rectifier Networks

We consider the application of LRP to deep neural networks with rectifier (ReLU) nonlinearities, arguably the most common choice in today's applications. It includes well-known architectures for image recognition such as VGG-16 [48] and Inception v3 [54], or neural networks used in reinforcement learning [35]. Deep rectifier networks are composed of neurons of the type:

$$a_k = \max\left(0, \textstyle\sum_{0,j} a_j w_{jk}\right). \tag{10.1}$$

The sum $\sum_{0,j}$ runs over all lower-layer activations $(a_j)_j$, plus an extra neuron representing the bias. More precisely, we set $a_0 = 1$ and define w_{0k} to be the neuron bias. We present three propagation rules for these networks and describe their properties.

Basic Rule (LRP-0) [7]. This rule redistributes in proportion to the contributions of each input to the neuron activation as they occur in Eq. (10.1):

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$$

This rule satisfies basic properties, such as $(a_j = 0) \vee (w_{j:} = 0) \Rightarrow R_j = 0$, which makes coincide concepts such as zero weight, deactivation, and absence of connection. Although this rule looks intuitive, it can be shown that a uniform application of this rule to the whole neural network produces an explanation that is equivalent to Gradient $\times$ Input (cf. [47]). As we have mentioned in the introduction, the gradient of a deep neural network is typically noisy, therefore one needs to design more robust propagation rules.

Epsilon Rule (LRP-ϵ) [7]. A first enhancement of the basic LRP-0 rule consists of adding a small positive term ϵ in the denominator:

Dampens the relevancy at each layer

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$$

The role of ϵ is to absorb some relevance when the contributions to the activation of neuron k are weak or contradictory. As ϵ becomes larger, only the most salient explanation factors survive the absorption. This typically leads to explanations that are sparser in terms of input features and less noisy.

Gamma Rule (LRP-γ). Another enhancement which we introduce here is obtained by favoring the effect of positive contributions over negative contributions:

This is the one with the transformer explanation paper

$$R_j = \sum_k \frac{a_j \cdot (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j \cdot (w_{jk} + \gamma w_{jk}^+)} R_k$$

The parameter γ controls by how much positive contributions are favored. As γ increases, negative contributions start to disappear. The prevalence of positive contributions has a limiting effect on how large positive and negative relevance can grow in the propagation phase. This helps to deliver more stable explanations. The idea of treating positive and negative contributions in an asymmetric manner was originally proposed in [7] with the LRP-$\alpha\beta$ rule (cf. Appendix 10.A). Also, choosing $\gamma \to \infty$ lets LRP-γ become equivalent to LRP-$\alpha_1\beta_0$ [7], the z^+-rule [36], and 'excitation-backprop' [59].

10.2.2 Implementing LRP Efficiently

The structure of LRP rules presented in Section 10.2.1 allows for an easy and efficient implementation. Consider the generic rule

$$R_j = \sum_k \frac{a_j \cdot \rho(w_{jk})}{\epsilon + \sum_{0,j} a_j \cdot \rho(w_{jk})} R_k, \tag{10.2}$$

of which LRP-$0/\epsilon/\gamma$ are special cases. The computation of this propagation rule can be decomposed in four steps:

$$\begin{aligned}
\forall_k: \; & z_k = \epsilon + \textstyle\sum_{0,j} a_j \cdot \rho(w_{jk}) && \text{(forward pass)} \\
\forall_k: \; & s_k = R_k / z_k && \text{(element-wise division)} \\
\forall_j: \; & c_j = \textstyle\sum_k \rho(w_{jk}) \cdot s_k && \text{(backward pass)} \\
\forall_j: \; & R_j = a_j \, c_j && \text{(element-wise product)}
\end{aligned}$$

The first step is a forward pass on a copy of the layer where the weights and biases have been applied the map $\theta \mapsto \rho(\theta)$, to which we further add the small

increment ϵ. The second and fourth steps are simple element-wise operations. For the third step, one notes that c_j can also be expressed as the gradient computation:

$$c_j = \left[\nabla\left(\sum_k z_k(\boldsymbol{a}) \cdot s_k\right)\right]_j$$

where $\boldsymbol{a} = (a_j)_j$ is the vector of lower-layer activations, where z_k is a function of it, and where s_k is instead treated as constant. This gradient can be computed via automatic differentiation, which is available in most neural networks libraries. In PyTorch[6], this propagation rule can be implemented by the following code:

```
def relprop(a,layer,R):
    z = epsilon + rho(layer).forward(a)
    s = R/(z+1e-9)
    (z*s.data).sum().backward()
    c = a.grad
    R = a*c
    return R
```

get the output of the forward pass

You divide here to do 2 things:
1. on the first backprop layer, it allows to get the gradient of `R` w.r.t the input such that the value of `R` ins't blown up.
2. On ever next backprop layer it essentially gets the value `c` for the downstream (previous) layer. Which is the gradient of `R` w.r.t. this layer's OUTPUT

This value should be equal to 1 for the first backward prop —>

get the gradient of the relevance wrt the input (aka how how input affects the relevancy)

Gradient showing how changes in input affect the relevancy (same length vector as input) —>

`(z*s.data)` is the value of `R`

<— is now the input * gradient of how much the relevancy changes w.r.t. the input

Essentially means how relevant the input is to the previous Relevancy score

The code is applicable to both convolution and dense layers with ReLU activation. The function "`rho`" returns a copy of the layer, where the weights and biases have been applied the map $\theta \mapsto \rho(\theta)$. The small additive term `1e-9` in the division simply enforces the behavior $0/0 = 0$. The operation "`.data`" lets the variable "`s`" become constant so that the gradient is not propagated through it. The function "`backward`" invokes the automatic differentiation mechanism and stores the resulting gradient in "`a`". Full code for the VGG-16 network is available at www.heatmapping.org/tutorial. When the structure of the neural network to analyze is more complex, or when we would like to compare and benchmark different explanation techniques, it can be recommended to use instead an existing software implementation such as iNNvestigate [1].

NOTE: Not doing `s.data` in `(z*s.data)` will result in 0 value gradients at `z` and thus at `a` as well and thus an `R` value of all 0s

10.2.3 LRP as a Deep Taylor Decomposition

Propagation rules of Section 10.2.1 can be interpreted within the Deep Taylor Decomposition (DTD) framework [36]. DTD views LRP as a succession of Taylor expansions performed locally at each neuron. More specifically, the relevance score R_k is expressed as a function of the lower-level activations $(a_j)_j$ denoted by the vector $\boldsymbol{a}$, and we then perform a first-order Taylor expansion of $R_k(\boldsymbol{a})$ at some reference point $\widetilde{\boldsymbol{a}}$ in the space of activations:

$$R_k(\boldsymbol{a}) = R_k(\widetilde{\boldsymbol{a}}) + \sum_{0,j}(a_j - \widetilde{a}_j) \cdot [\nabla R_k(\widetilde{\boldsymbol{a}})]_j + \ldots \tag{10.3}$$

First-order terms (summed elements) identify how much of R_k should be redistributed on neurons of the lower layer. Due to the potentially complex relation between $\boldsymbol{a}$ and R_k, finding an appropriate reference point and computing the gradient locally is difficult.

The divide `R` by `z` (to get `s`) makes sense because of the following:
In the first back-propagation, you do the element wise divide of `R` (the one hot vector of the class) by the output (`z`) (logits). When you do the '.backward()' pass (`torch.autograd.grad(x1_x_w1, x1, S1)[0]`), it will get you the gradient of the Relevance (`R`) w.r.t. `z`, but scaled such that the resulting Relevance for that first layer is scaled to not explode. (Also, this division value (`s`) IS the value gradient for `z`)
Recall that the Relevance = (input)*(the gradient of R w.r.t. the input), so on the next layer you backprop through, the `R/z` is essentially the gradient of previous (downstream) layer w.r.t. `R`. So now when you do '.backward()' for this subsequent layer, for the `(z*s)` part, you're getting the gradient of how that layer you're currently on impacts `R`. `R/z` will set the gradient of the current layer's output, `z` be the value of `c` in the previous (downstream) layer, which makes total sense.

As 's' is the gradient (of input, `a` w.r.t. `R`) of the next downstream layer. So that your current layer's output gradient is the gradient of the next layer's input w.r.t. to R, which makes total sense as that current layer's output is the downstream layer's input.

Relevance Model. In order to obtain a closed-form expression for the terms of Eq. (10.3), one needs to substitute the true relevance function $R_k(\boldsymbol{a})$ by a relevance model $\widehat{R}_k(\boldsymbol{a})$ that is easier to analyze [36]. One such model is the modulated ReLU activation:

$$\widehat{R}_k(\boldsymbol{a}) = \max\left(0, \textstyle\sum_{0,j} a_j w_{jk}\right) \cdot c_k.$$

The modulation term c_k is set constant and in a way that $\widehat{R}_k(\boldsymbol{a}) = R_k(\boldsymbol{a})$ at the current data point. Treating c_k as constant can be justified when R_k results from application of LRP-$0/\epsilon/\gamma$ in the higher layers (cf. Appendix 10.B). A Taylor expansion of the relevance model $\widehat{R}_k(\boldsymbol{a})$ on the activation domain gives:

$$\widehat{R}_k(\boldsymbol{a}) = \widehat{R}_k(\widetilde{\boldsymbol{a}}) + \textstyle\sum_{0,j}(a_j - \widetilde{a}_j) \cdot w_{jk}\, c_k.$$

Second- and higher-order terms are zero due to the linearity of the ReLU function on its activated domain. The zero-order term can also be made arbitrarily small by choosing the reference point near the ReLU hinge. Once a reference point is chosen, first-order terms can be easily computed, and redistributed to neurons in the lower layer. Figure 10.3 (a-c) illustrates how deep Taylor decomposition is applied at a given neuron.

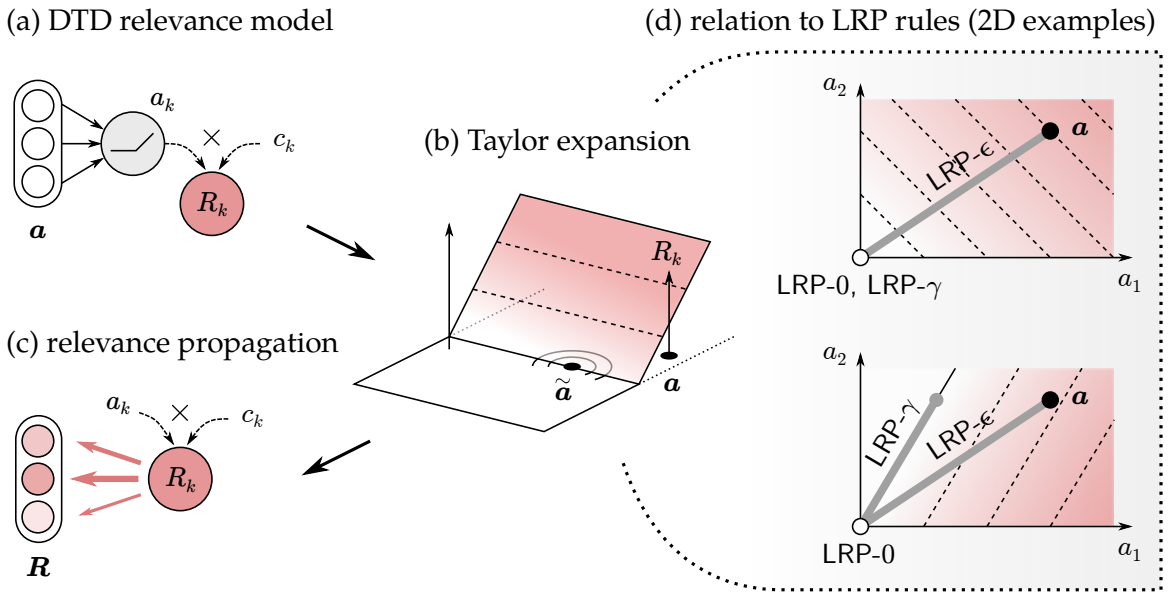

Fig. 10.3. Illustration of DTD: (a) graph view of the relevance model, (b) function view of the relevance model and reference point at which the Taylor expansion is performed, (c) propagation of first-order terms on the lower layer.

Relation to LRP-$0/\epsilon/\gamma$. Each choice of reference point $\widetilde{\boldsymbol{a}}$ leads to a different way of redistributing relevance. Interestingly, specific choices of reference points

reduce to the LRP propagation rules defined in Section 10.2.1. LRP-0 is recovered by choosing $\widetilde{\boldsymbol{a}} = \mathbf{0}$. LRP-$\epsilon$ is recovered by choosing $\widetilde{\boldsymbol{a}} = \epsilon \cdot (a_k + \epsilon)^{-1}\, \boldsymbol{a}$. LRP-$\gamma$ is recovered by choosing $\widetilde{\boldsymbol{a}}$ at the intersection between the ReLU hinge and the line $\{\boldsymbol{a} - t \cdot \boldsymbol{a} \odot (\mathbf{1} + \gamma \cdot \mathbf{1}_{\boldsymbol{w}_k \succeq \mathbf{0}}) \mid t \in \mathbb{R}\}$, where $\mathbf{1}_{\{\cdot\}}$ is an indicator function applied element-wise. The relation between LRP and DTD root points is further illustrated on simple two-dimensional neurons in Fig. 10.3 (d). For all three LRP propagation rules, one can show that the DTD reference points always satisfy $\widetilde{\boldsymbol{a}} \succeq \mathbf{0}$, and therefore match the domain of ReLU activations received as input [36]. A further property of reference points one can look at is the distance $\|\widetilde{\boldsymbol{a}} - \boldsymbol{a}\|$. The smaller the distance, the more contextualized the explanation will be, and the lower the number of input variables that will appear to be in contradiction. LRP-0 has the highest distance. LRP-ϵ and LRP-γ reduce this distance significantly.

10.3 Which LRP Rule for Which Layer?

As a general framework for propagation, LRP leaves much flexibility on which rule to use at each layer, and how the parameters ϵ and γ should be set. Selecting LRP parameters optimally would require a measure of explanation quality. How to assess explanation quality is still an active research topic [43, 16, 40, 38], and a full discussion is beyond the scope of this chapter. Instead, we discuss LRP in the light of two general and well-agreed desirable properties of an explanation: *fidelity* and *understandability* [52]. In other words, an explanation should be an accurate representation of the output neuron of interest, and it should also be easy to interpret for a human. Note that to visually assess the fidelity of an explanation, one needs to assume that the network has solved the task in a "ground-truth" manner, i.e. using the correct visual features to support its prediction, and ignoring distracting factors in the image.

Figure 10.4 shows for a given input image (of size 224×224), various LRP explanations of the VGG-16 [48] output neuron 'castle'. These explanations are either obtained by uniform application of a single propagation rule at all layers, or by a composite strategy [29] where different rules are used at different layers.

We observe strong differences in the explanations. *Uniform LRP-0* picks many local artifacts of the function. The explanation is overly complex and does not focus sufficiently on the actual castle in the image. The explanation is neither faithful nor understandable. *Uniform LRP-ϵ* removes noise elements in the explanation to keep only a limited number features that match the actual castle in the image. It is a faithful explanation, but too sparse to be easily understandable. *Uniform LRP-γ* is easier for a human to understand because features are more densely highlighted, but it also picks unrelated concepts such as the lamp post, making it unfaithful. *Composite LRP* overcomes the disadvantages of the approaches above. The features of the castle are correctly identified and fully highlighted, thereby making the explanation both faithful and understandable.

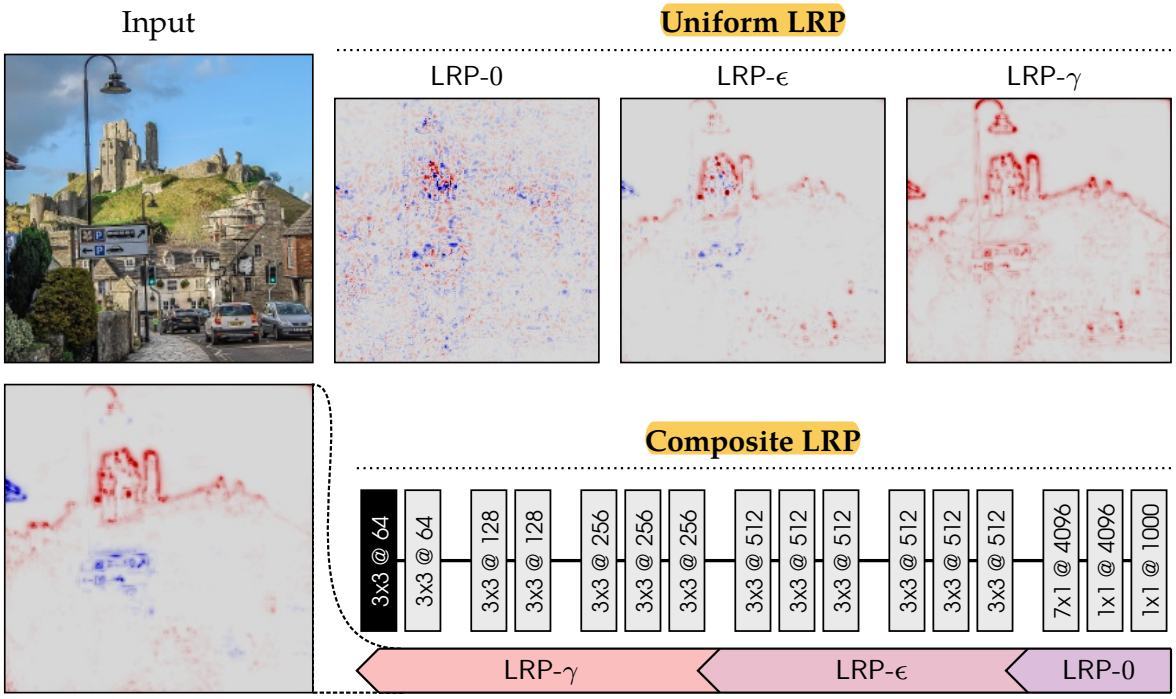

Fig. 10.4. Input image and pixel-wise explanations of the output neuron 'castle' obtained with various LRP procedures. Parameters are $\epsilon = 0.25\,\text{std}$ and $\gamma = 0.25$.

The reason why Composite LRP delivers a better explanation can be traced to the qualitative differences between the various layers of the VGG-16 neural network:

Upper layers have only approximately 4 000 neurons (i.e. on average 4 neurons per class), making it likely that the many concepts forming the different classes are entangled. Here, a propagation rule close to the function and its gradient (e.g. LRP-0) will be insensitive to these entanglements.

Middle layers have a more disentangled representation, however, the stacking of many layers and the weight sharing in convolutions introduces spurious variations. LRP-ϵ filters out these spurious variations and retains only the most salient explanation factors.

Lower layers are similar to middle layers, however, LRP-γ is more suitable here, as this rule tends to spread relevance uniformly to the whole feature rather than capturing the contribution of every individual pixel. This makes the explanation more understandable for a human.

Overall, in order to apply LRP successfully on a new task, it is important to carefully inspect the properties of the neural network layers, and to ask the human what kind of explanation is most understandable for him.

10.3.1 Handling the Top Layer

The quantity we have explained so far is the score z_c for class c, computed from lower-layer activations $(a_k)_k$ as:

$$z_c = \sum_{0,k} a_k w_{kc}.$$

It is linked to the predicted class probability via the softmax function $\mathrm{P}(\omega_c) = \exp(z_c)/\sum_{c'} \exp(z_{c'})$. Fig. 10.5 (middle) shows an explanation of the score $z_{\text{passenger\_car}}$ for some image containing a locomotive, a passenger car and other elements in the background. The explanation retains the passenger car features, but also features of the locomotive in front of it. This shows that the quantity z_c is not truly selective for the class to explain.

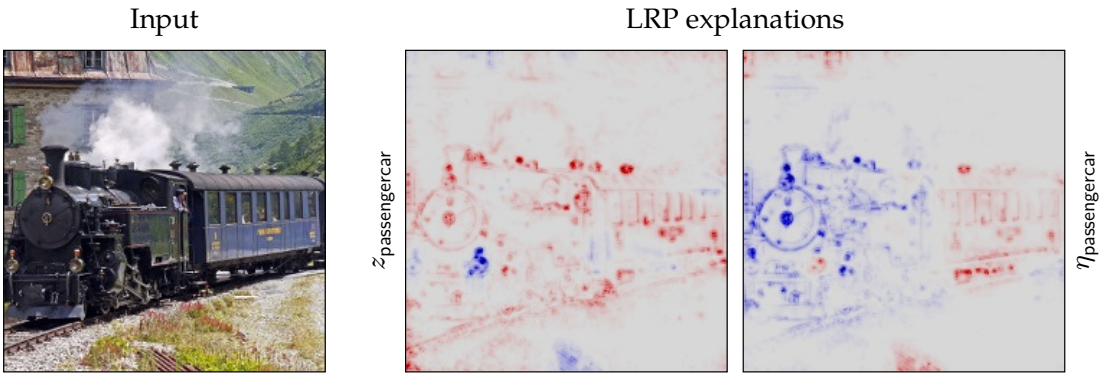

Fig. 10.5. Explanations obtained for the output neuron 'passenger_car' and for the actual probability of the class 'passenger_car'. The locomotive switches from positive to negatively relevant.

Alternately, we can opt for explaining $\eta_c = \log[\mathrm{P}(\omega_c)/(1-\mathrm{P}(\omega_c))]$, which can be expressed by the sequence of layers:

$$z_{c,c'} = \sum_{0,k} a_k (w_{kc} - w_{kc'})$$
$$\eta_c = -\log \sum_{c' \neq c} \exp(-z_{c,c'}).$$

The first layer represents the log-probability ratios $\log[\mathrm{P}(\omega_c)/\mathrm{P}(\omega_{c'})]$, and the second layer performs a reverse log-sum-exp pooling over these ratios. A propagation rule for this type of pooling layer was proposed in [26]: Relevance is redistributed on the pooled neurons following a min-take-most strategy: $R_{c,c'} = z_{c,c'} \cdot \exp(-z_{c,c'})/\sum_{c'' \neq c} \exp(-z_{c,c''})$. These scores can then be further propagated into the neural network with usual LRP rules. Figure 10.5 (right) shows the explanation for $\eta_{\text{passenger\_car}}$. Positive evidence becomes sparser, and the locomotive turns blue (i.e. negatively relevant). This reflects the fact that the presence of the locomotive in the image raises the probability for the class 'locomotive' and thus lowers it for the class 'passenger_car'.

10.3.2 Handling Special Layers

Practical neural networks are often equipped with special layers that facilitate optimization, incorporate some predefined invariance into the model, or handle a particular type of input data. We briefly review how to handle some of these layers within the LRP framework.

Spatial Pooling Layers are often used between convolution layers to promote local translation invariance in the model. A sum-pooling layer applied to positive activations can be easily rewritten as a standard linear-ReLU layer. Thus all LRP rules we have presented here can also be applied to sum-pooling layers. Max-pooling layers, on the other hand, can either be handled by a winner-take-all redistribution scheme [7], or by using the same rules as for the sum-pooling case [36, 37]. In this chapter, we have used the second option.

Batch Normalization Layers are commonly used to facilitate training and improve prediction accuracy. At test time, they simply consist of a centering and rescaling operation. These layers can therefore be absorbed by the adjacent linear layer without changing the function. This allows to recover the canonical neural network structure needed for applying LRP. Aka do nothing for batch norm layers

Input Layers are different from intermediate layers as they do not receive ReLU activations as input but pixels or real values. Special rules for these layers can also be derived from the DTD framework [36] (cf. Appendix 10.A). In this chapter, we made use of the $z^{\mathcal{B}}$-rule, which is suitable for pixels.

10.4 LRP Beyond Deep Networks

Deep neural networks have been particularly successful on tasks involving classification and regression. Other problems such as unsupervised modeling, time series forecasting, and pairwise matching, have been traditionally handled by other types of models. Here, we discuss various extensions that let LRP be applied to this broader class of models.

Unsupervised Models. Unsupervised learning algorithms extract structures from unlabeled data from which properties such as membership to some cluster or degree of anomaly can be predicted. In order to explain these predictions, a novel methodology called Neuralization-Propagation (NEON) was proposed [25, 26]: The learned unsupervised model is first 'neuralized' (i.e. transformed into a functionally equivalent neural network). Then, an LRP procedure is built in order to propagate the prediction backward in the neural network.

Ehhh this is ok. Acceptable over all population data, but that's *impossible to have

In one-class SVMs [44], predicted anomaly could be rewritten as a min-pooling over support vector distances [25]. Similarly, in k-means, predicted cluster membership could be rewritten as pooling over local linear discriminants

between competing clusters [26]. For each extracted neural network, suitable LRP rules could be designed based on the DTD methodology. Overall, the proposed Neuralization-Propagation approach endows these unsupervised models with fast and reliable explanations.

Time Series Prediction. To predict the next steps of a time series, one must ideally be able to identify the underlying dynamical system and simulate it forward. A popular model for this is the LSTM [22]. It uses product interactions of the type

$$h_k = \text{sigm}(\textstyle\sum_j a_j v_{jk} + c_k) \cdot g\big(\sum_j a_j w_{jk} + b_k\big).$$

The first term is a gate that regulates how the signal is transferred between the internal state and the real-world. The second term is the signal itself. A successful strategy for applying LRP in these models is to let all relevance flow through the second term [6, 40, 42, 56]. Furthermore, when g is chosen to be a ReLU function, and if the gating function is strictly positive or locally constant, this strategy can also be justified within the DTD framework.

Pairwise Matching. A last problem for which one may require explanations is when predicting if two vectors $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ match. This problem arises, for example, when modeling the relation between an image and a transformed version of it [34], or in recommender systems, when modeling the relation between users and products [55]. An approach to pairwise matching is to build product neurons of the type $a_k = \max(0, \sum_i x_i w_{ik}) \cdot \max(0, \sum_j y_j v_{jk})$. A propagation rule for this product of neurons is given by [31]:

$$R_{ij} = \sum_k \frac{x_i y_j w_{ik} v_{jk}}{\sum_{ij} x_i y_j w_{ik} v_{jk}} R_k.$$

This propagation rule can also be derived from DTD when considering second-order Taylor expansions. The resulting explanation is in terms of pairs of input features i and j from each modality.

10.5 Conclusion

We have reviewed Layer-wise Relevance Propagation (LRP), a technique that can explain the predictions of complex state-of-the-art neural networks in terms of input features, by propagating the prediction backward in the network by means of propagation rules. LRP has a number of properties that makes it attractive: Propagation rules can be implemented efficiently and modularly in most modern neural network software and a number of these rules are furthermore embeddable in the Deep Taylor Decomposition framework. Parameters of the LRP rules can be set in a way that high explanation quality is obtained even for complex models. Finally, LRP is extensible beyond deep neural network classifiers to a broader range of machine learning models and tasks. This makes

it applicable to a large number of practical scenarios where explanation is needed.

Acknowledgements. This work was supported by the German Ministry for Education and Research as Berlin Big Data Centre (01IS14013A), Berlin Center for Machine Learning (01IS18037I) and TraMeExCo (01IS18056A). Partial funding by DFG is acknowledged (EXC 2046/1, project-ID: 390685689). This work was also supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (No. 2017-0-00451, No. 2017-0-01779).

Appendices

10.A List of Commonly Used LRP Rules

The table below gives a non-exhaustive list of propagation rules that are commonly used for explaining deep neural networks with ReLU nonlinearities. The last column in the table indicates whether the rules can be derived from the deep Taylor decomposition [36] framework.

This is the method the transformer paper uses

Name	Formula	Usage	DTD
LRP-0 [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$	upper layers	✓
LRP-ϵ [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$	middle layers	✓
LRP-γ	$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k$	lower layers	✓
LRP-$\alpha\beta$ [7]	$R_j = \sum_k \left(\alpha \frac{(a_j w_{jk})^+}{\sum_{0,j} (a_j w_{jk})^+} - \beta \frac{(a_j w_{jk})^-}{\sum_{0,j} (a_j w_{jk})^-} \right) R_k$	lower layers	$\times^\star$
flat [30]	$R_j = \sum_k \frac{1}{\sum_j 1} R_k$	lower layers	$\times$
w^2-rule [36]	$R_i = \sum_j \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j$	first layer ($\mathbb{R}^d$)	✓
$z^\mathcal{B}$-rule [36]	$R_i = \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j$	first layer (pixels)	✓

(* DTD interpretation only for the case $\alpha = 1, \beta = 0$.)

Here, we have used the notation $(\cdot)^+ = \max(0, \cdot)$ and $(\cdot)^- = \min(0, \cdot)$. For the LRP-$\alpha\beta$ rule, the parameters α, β are subject to the conservation constraint $\alpha = \beta + 1$. For the $z^\mathcal{B}$-rule the parameters l_i, h_i define the box constraints of the input domain ($\forall_i : \; l_i \leq x_i \leq h_i$).

10.B Justification of the Relevance Model

We give here a justification similar to [36, 37] that the relevance model $\widehat{R}_k(\boldsymbol{a})$ of Section 10.2.3 is suitable when relevance R_k results from applying LRP-$0/\epsilon/\gamma$ in the higher layers. The generic propagation rule

$$R_k = \sum_l \frac{a_k \cdot \rho(w_{kl})}{\epsilon + \sum_{0,k} a_k \cdot \rho(w_{kl})} R_l,$$

of which LRP-$0/\epsilon/\gamma$ are special cases, can be rewritten as $R_k = a_k c_k$ with

$$c_k(\boldsymbol{a}) = \sum_l \rho(w_{kl}) \, \frac{\max\left(0, \sum_{0,k} a_k(\boldsymbol{a}) \cdot w_{kl}\right)}{\epsilon + \sum_{0,k} a_k(\boldsymbol{a}) \cdot \rho(w_{kl})} \, c_l(\boldsymbol{a}),$$

where the dependences on lower activations $\boldsymbol{a}$ have been made explicit. Assume $c_l(\boldsymbol{a})$ to be approximately locally constant w.r.t. $\boldsymbol{a}$. Because other terms that depend on $\boldsymbol{a}$ are diluted by two nested sums, it is plausible that $c_k(\boldsymbol{a})$ is again locally approximately constant, which is the assumption made by the relevance model $\widehat{R}_k(\boldsymbol{a})$.

References

1. Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K.T., Montavon, G., Samek, W., Müller, K.R., Dähne, S., Kindermans, P.J.: iNNvestigate neural networks!. Journal of Machine Learning Research **20**(93), 1–8 (2019)
2. Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ...: Deep Speech 2 : End-to-end speech recognition in English and Mandarin. In: Proceedings of the 33nd International Conference on Machine Learning. pp. 173–182 (2016)
3. Anders, C., Montavon, G., Samek, W., Müller, K.R.: Understanding patch-based learning of video data by explaining predictions. In: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Lecture Notes in Computer Science **11700**, Springer (2019)
4. Arbabzadah, F., Montavon, G., Müller, K., Samek, W.: Identifying individual facial expressions by deconstructing a neural network. In: 38th German Conference on Pattern Recognition. pp. 344–354 (2016)
5. Arras, L., Horn, F., Montavon, G., Müller, K.R., Samek, W.: "What is relevant in a text document?": An interpretable machine learning approach. PLoS ONE **12**(8), e0181142 (2017)
6. Arras, L., Montavon, G., Müller, K.R., Samek, W.: Explaining recurrent neural network predictions in sentiment analysis. In: Proceedings of the 8th EMNLP Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. pp. 159–168 (2017)
7. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**(7), e0130140 (2015)
8. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.: How to explain individual classification decisions. Journal of Machine Learning Research **11**, 1803–1831 (2010)

9. Baldi, P., Sadowski, P., Whiteson, D.: Searching for exotic particles in high-energy physics with deep learning. Nature Communications **5**(1) (jul 2014)
10. Balduzzi, D., Frean, M., Leary, L., Lewis, J.P., Ma, K.W., McWilliams, B.: The shattered gradients problem: If resnets are the answer, then what is the question? In: Proceedings of the 34th International Conference on Machine Learning. pp. 342–350 (2017)
11. Bazen, S., Joutard, X.: The Taylor decomposition: A unified generalization of the Oaxaca method to nonlinear models. Working papers, HAL (2013)
12. Binder, A., Bockmayr, M., Hägele, M., Wienert, S., Heim, D., Hellweg, K., Stenzinger, A., Parlow, L., Budczies, J., Goeppert, B., Treue, D., Kotani, M., Ishii, M., Dietel, M., Hocke, A., Denkert, C., Müller, K., Klauschen, F.: Towards computational fluorescence microscopy: Machine learning-based integrated prediction of morphological and molecular tumor profiles. CoRR **abs/1805.11178** (2018)
13. Calude, C.S., Longo, G.: The deluge of spurious correlations in big data. Foundations of Science **22**(3), 595–612 (2017)
14. Chmiela, S., Tkatchenko, A., Sauceda, H.E., Poltavsky, I., Schütt, K.T., Müller, K.R.: Machine learning of accurate energy-conserving molecular force fields. Science Advances **3**(5), e1603015 (may 2017)
15. Clark, P., Matwin, S.: Using qualitative models to guide inductive learning. In: Proceedings of the 10th International Conference on Machine Learning. pp. 49–56 (1993)
16. Doshi-Velez, F., Kim, B.: Considerations for Evaluation and Generalization in Interpretable Machine Learning, pp. 3–17. Springer International Publishing (2018)
17. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115–118 (2017)
18. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: IEEE International Conference on Computer Vision. pp. 3449–3457 (2017)
19. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research **3**(Mar), 1157–1182 (2003)
20. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web. pp. 173–182 (2017)
21. Hettwer, B., Gehrer, S., Güneysu, T.: Deep neural network attribution methods for leakage analysis and symmetric key recovery. IACR Cryptology ePrint Archive **2019**, 143 (2019)
22. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
23. Hochuli, J., Helbling, A., Skaist, T., Ragoza, M., Koes, D.R.: Visualizing convolutional neural network protein-ligand scoring. Journal of Molecular Graphics and Modelling **84**, 96–108 (sep 2018)
24. Horst, F., Lapuschkin, S., Samek, W., Müller, K.R., Schöllhorn, W.I.: Explaining the unique nature of individual gait patterns with deep learning. Scientific Reports **9**, 2391 (feb 2019)
25. Kauffmann, J., Müller, K.R., Montavon, G.: Towards explaining anomalies: A deep Taylor decomposition of one-class models. CoRR **abs/1805.06230** (2018)
26. Kauffmann, J., Esders, M., Montavon, G., Samek, W., Müller, K.R.,: From clustering to cluster explanations via neural networks. CoRR **abs/1906.07633** (2019)

27. Landecker, W., Thomure, M.D., Bettencourt, L.M.A., Mitchell, M., Kenyon, G.T., Brumby, S.P.: Interpreting individual classifications of hierarchical networks. In: IEEE Symposium on Computational Intelligence and Data Mining. pp. 32–38 (2013)
28. Lapuschkin, S., Binder, A., Montavon, G., Müller, K.R., Samek, W.: Analyzing classifiers: Fisher vectors and deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2912–2920 (2016)
29. Lapuschkin, S., Binder, A., Müller, K.R., Samek, W.: Understanding and comparing deep neural networks for age and gender classification. In: IEEE International Conference on Computer Vision Workshops. pp. 1629–1638 (2017)
30. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.R.: Unmasking Clever Hans predictors and assessing what machines really learn. Nature Communications **10**, 1096 (2019)
31. Leupold, S.: Second-order Taylor decomposition for explaining spatial transformation of images. Master's thesis, Technische Universität Berlin (2017)
32. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. pp. 50–56 (2016)
33. Mayr, A., Klambauer, G., Unterthiner, T., Hochreiter, S.: DeepTox: Toxicity prediction using deep learning. Frontiers in Environmental Science **3**, 80 (2016)
34. Memisevic, R., Hinton, G.E.: Learning to represent spatial transformations with factored higher-order Boltzmann machines. Neural Computation **22**(6), 1473–1492 (2010)
35. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (feb 2015)
36. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognition **65**, 211–222 (2017)
37. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
38. Narayanan, M., Chen, E., He, J., Kim, B., Gershman, S., Doshi-Velez, F.: How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. CoRR **abs/1802.00682** (2018)
39. Perotin, L., Serizel, R., Vincent, E., Guérin, A.: CRNN-based multiple DoA estimation using acoustic intensity features for ambisonics recordings. J. Sel. Topics Signal Processing **13**(1), 22–33 (2019)
40. Poerner, N., Schütze, H., Roth, B.: Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. pp. 340–350 (2018)
41. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144 (2016)
42. Rieger, L., Chormai, P., Montavon, G., Hansen, L.K., Müller, K.R.: Structuring neural networks for more explainable predictions. In: Explainable and Interpretable Models in Computer Vision and Machine Learning, pp. 115–131. Springer International Publishing (2018)

43. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R.: Evaluating the visualization of what a deep neural network has learned. IEEE Transactions on Neural Networks and Learning Systems **28**(11), 2660–2673 (2017)
44. Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt, J.C.: Support vector method for novelty detection. In: Advances in Neural Information Processing Systems 12. pp. 582–588 (1999)
45. Schütt, K.T., Arbabzadah, F., Chmiela, S., Müller, K.R., Tkatchenko, A.: Quantum-chemical insights from deep tensor neural networks. Nature Communications **8**, 13890 (jan 2017)
46. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: Proceedings of the 34th International Conference on Machine Learning. pp. 3145–3153 (2017)
47. Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A.: Not just a black box: Learning important features through propagating activation differences. CoRR **abs/1605.01713** (2016)
48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (2015)
49. Smilkov, D., Thorat, N., Kim, B., Viégas, F.B., Wattenberg, M.: Smoothgrad: removing noise by adding noise. CoRR **abs/1706.03825** (2017)
50. Sturm, I., Lapuschkin, S., Samek, W., Müller, K.R.: Interpretable deep neural networks for single-trial EEG classification. Journal of Neuroscience Methods **274**, 141–145 (2016)
51. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning. pp. 3319–3328 (2017)
52. Swartout, W.R., Moore, J.D.: Second generation expert systems. chap. Explanation in Second Generation Expert Systems, pp. 543–585. Springer-Verlag New York, Inc. (1993)
53. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: 2nd International Conference on Learning Representations (2014)
54. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
55. Xue, H., Dai, X., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 3203–3209 (2017)
56. Yang, Y., Tresp, V., Wunderle, M., Fasching, P.A.: Explaining therapy predictions with layer-wise relevance propagation in neural networks. In: IEEE International Conference on Healthcare Informatics. pp. 152–162 (2018)
57. Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: Attacks and defenses for deep learning. IEEE Transactions on Neural Networks and Learning Systems pp. 1–20 (2019)
58. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proc. of European Conference on Computer Vision (ECCV). pp. 818–833. Springer (2014)
59. Zhang, J., Bargal, Sarah Adeland Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. International Journal of Computer Vision **126**(10), 1084–1102 (2018)

60. Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M.: Visualizing deep neural network decisions: Prediction difference analysis. In: International Conference on Learning Representations (2017)