Methods for interpreting and understanding deep neural networks

Abstract

This paper provides an entry point to the problem of interpreting a deep neural network model and explaining its predictions.
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

Introduction

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.
It has been a common belief, that simple models provide higher interpretability than complex ones.
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.

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.
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.

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”.
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 variable
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.

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.

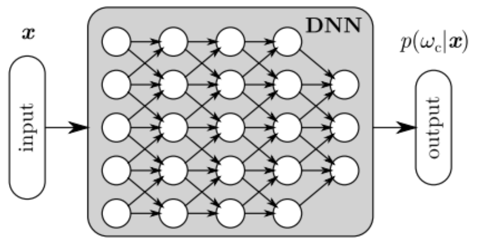
Examples of domains that are interpretable are images (arrays of pixels), or texts (sequences of words).
A human can look at them and read them respectively.

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.

Interpreting a DNN model

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.
The neurons of the network jointly implement a complex nonlinear mapping from the input to the output.
The concept that must be interpreted is usually represented by a neuron in the top layer.
Building the prototype can be formulated within the activation maximization framework.



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].

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

Improving AM with an expert

In order to obtain more meaningful prototypes, the ℓ2-regularizer can be replaced by a more sophisticated one [44,52] called “expert”.

The expert can be, for example, a model p(x) of the data.

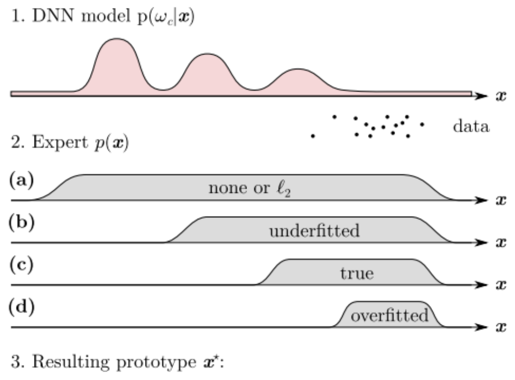
This leads to the new optimization problem:

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

The prototype x obtained by solving this optimization problem will simultaneously produce strong class response and resemble the data.

A possible choice for the expert is the Gaussian RBM [25]. It can represent complex distributions and has a gradient in the input domain.

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



On one extreme a coarse expert (a) reduces the optimization problem to the maximization of the class probability function p(ωc |x).

Performing AM in code space

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

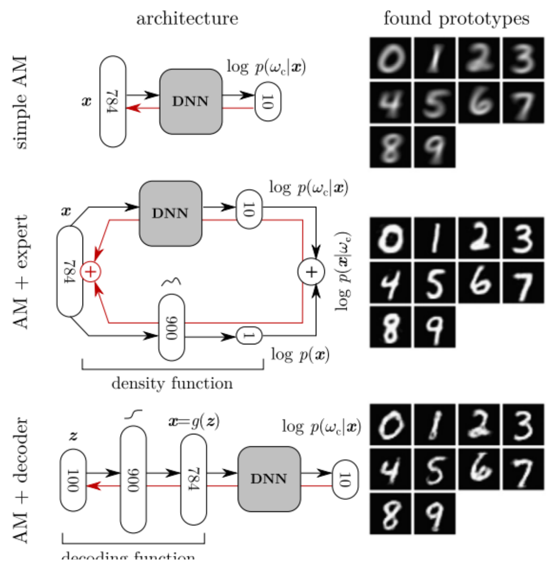
1. Sample from a simple distribution q(z) ~ N (0, I) defined in some abstract code space Z.
2. Apply to the sample a decoding function g : Z → X , that maps it back to the original input domain.

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.

Nguyen et al. [51] proposed to build a prototype for ωc by incorporating such generative model in the activation maximization framework.
The optimization problem is redefined as:

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

favoring codes with high probability.
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.



decoding function

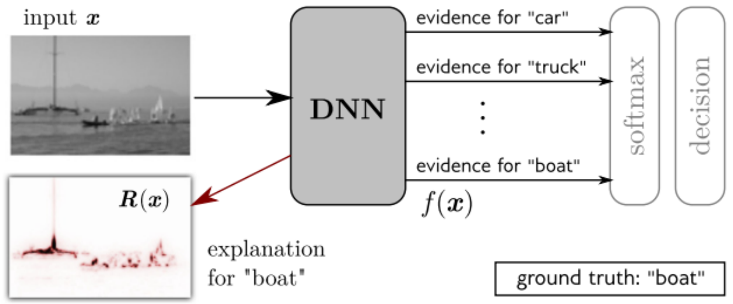
Each prototype is classified with full certainty by the DNN.
However, only with an expert or a decoding function, the prototypes become sharp and realistic-looking

From global to local analysis

When considering complex machine learning problems, probability functions p(ωc |x) and p(x) might be multimodal or strongly elongated, so that no single prototype x fully represents the modeled concept ωc.
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 ωc using multiple local prototypes instead of a single global one.
Producing an exhaustive description of the modeled concept ωc is however not always necessary.
An expedient way of introducing locality into the analysis would be to add a localization term η · x − x02 to the AM objective, where x0 is a reference point.
Instead, when trying to interpret the concept ωc locally, a more relevant question to ask is “what features of x make it representative of the concept ωc?”.

Explaining DNN decisions

In this section, we ask for a given data point x, what makes it representative of a certain concept ωc encoded at the output of the deep neural network (DNN).
The output neuron that encodes this concept can be described as a function f (x) of the input.
A common approach to explanation is to view the data point x as a collection of features (xi) d i=1, and to assign to each of these, a score Ri determining how relevant the feature xi is for explaining f (x).



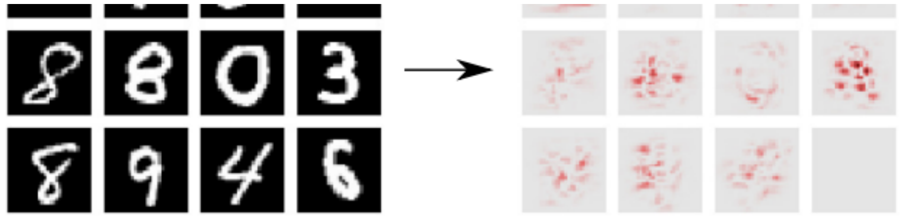
Sensitivity analysis

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

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

The most relevant input features are those to which the output is most sensitive.

input
sensitivity analysis



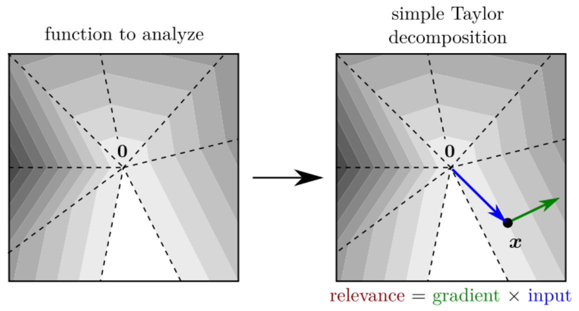
Thus, sensitivity analysis does not provide an explanation of the function value f (x), but of its local slope.
In the example above, the heatmap indicates what pixels make the digit belong more/less to the target class rather than what makes the digit belong to that class.

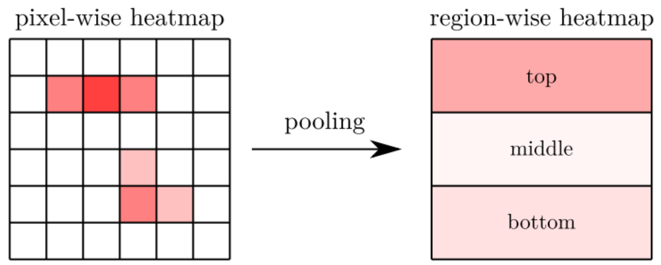
Simple Taylor decomposition

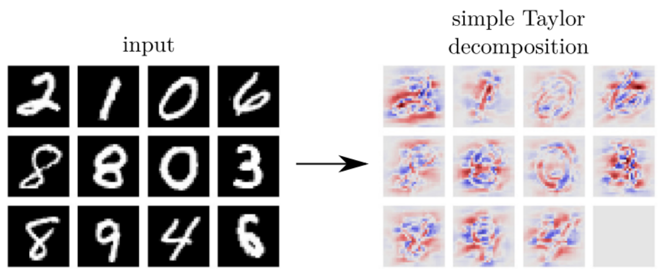
$$f(\boldsymbol{x}) = \sum_{i=1}^{d} R_i(\boldsymbol{x}) + O(\boldsymbol{x}\boldsymbol{x}^\top)$$

where the relevance scores

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





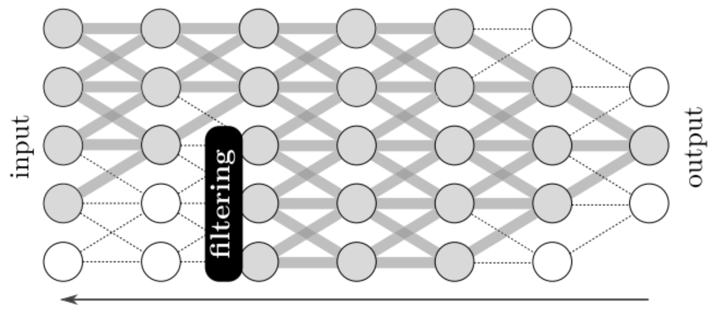


It can be observed that heatmaps are more complete than those obtained by sensitivity analysis.
However, they are characterized by a large amount of negative relevance, here 38% of the total relevance.

Backward propagation techniques

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



Non-conserving backward propagation techniques include deconvolution [74], and its extension guided backprop [63]
Unlike LRP and other conserving techniques, the visualization produced by these methods cannot directly be interpreted as relevance scores

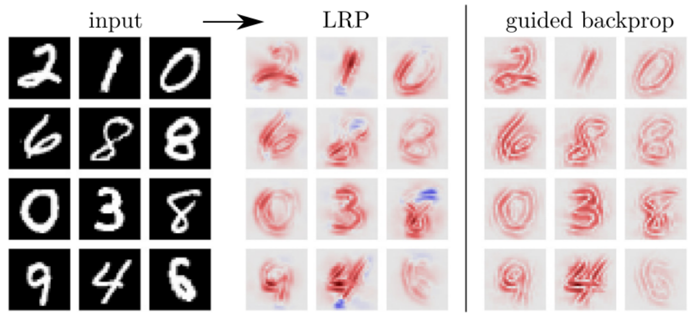
In comparison to the simple gradient-based methods of Sections 4.1 and 4.2, backward propagation techniques such as LRP or deconvolution were shown empirically to scale better to complex DNN models
These techniques provide a further practical advantage: The quantity being propagated can be filtered to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

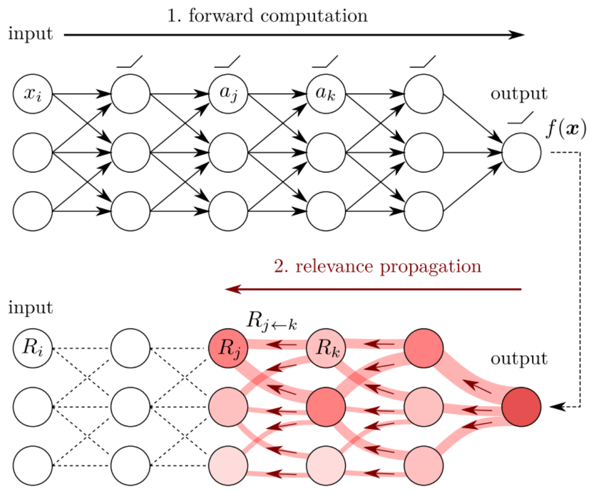
Filtering can be useful, for example, in the context of multitask learning, where some hidden neurons are specific to a given task and other neurons are shared.
(1) what is specifically relevant to the given task
(2) what is commonly relevant to all tasks

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]	✓	✓

Layer-wise relevance propagation (LRP)



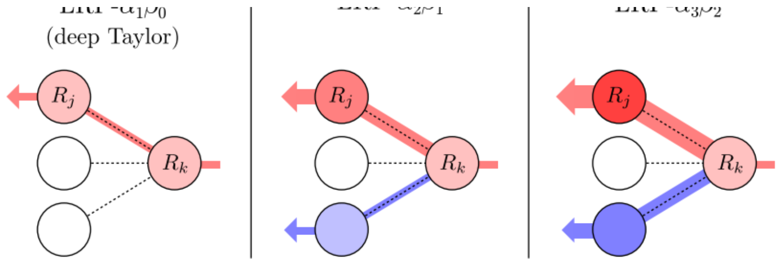


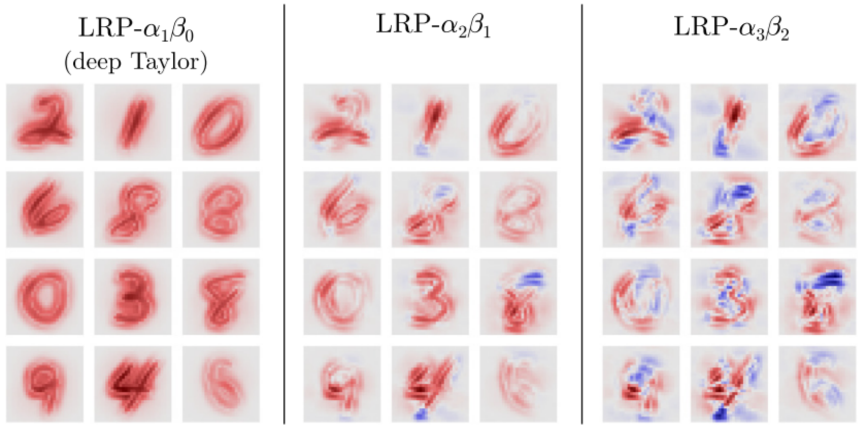
LRP technique is rooted in a conservation principle, where each neuron receives a share of the network output, and redistributes it to its predecessors in equal amount, until the input variables are reached [5,34].

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.

LRP propagation rules

LRP-$\alpha_1\beta_0$ | LRP-$\alpha_2\beta_1$ | LRP-$\alpha_3\beta_2$

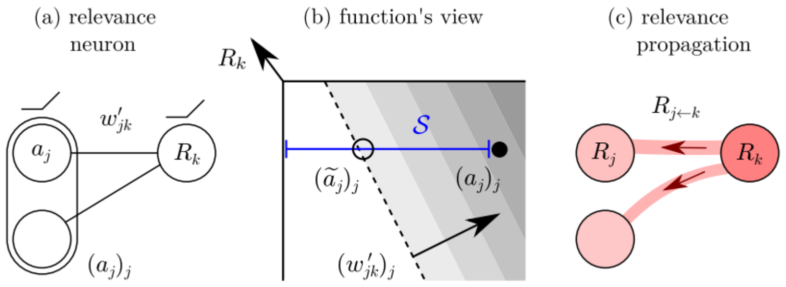




When α = 1, the heatmaps contain only positive relevance, and the latter is spread along the contour of the digits in a fairly uniform manner
When choosing α = 2, some regions in the images become negatively relevant (e.g. the broken upper-loop of the last digit “8”), but the negative relevance still amounts to only 5% of the total relevance.
When setting the higher value α = 3, negative relevance starts to appear in a seemingly random fashion, with the share of total relevance surging to 30%

LRP and deep Taylor decomposition

The analysis relies on a special structure of the relevance scores Rk at each layer, which have to be the product of the corresponding neuron activations and positive constant terms.



Handling special layers

To ensure a broader applicability for LRP, other layer types need to be considered:

Input layer.

Deep Taylor decomposition adapts to the new input domain by modifying the root search direction S to remain inside of it

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

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

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

A possible propagation rule for this layer is

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

This rule selects the most relevant neurons in the pool, while also ensuring a continuous transition in the space of pooled activations.

Other layers.

a variety of layers and architectures are used to handle specific types of signals or to reach maximum performance.

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.

Recommendations and tricks

Structuring the DNN for maximum explainability

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.

The second recommendation focuses on spatial or temporal pooling: Use sum-pooling layers abundantly, and prefer them to other types of pooling layers.

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 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

Choosing the LRP rules for explanation

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

If negative relevance is needed, or the heatmaps are too diffuse, replace the rule LRP-α1β0 by LRP-α2β1 in the hidden layers.
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.

Tricks for implementing LRP

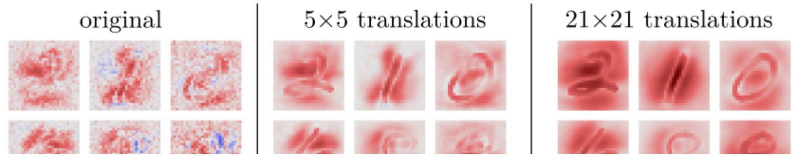
$$R_j = a_j \sum_k \frac{w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k,$$

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)

```
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
```

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



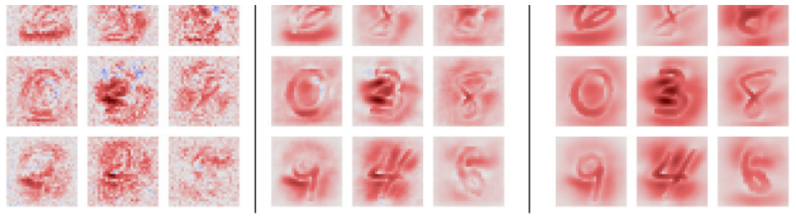


Fig. 15. Heatmaps obtained by explaining a fully-connected DNN on MNIST with LRP-$\alpha_2\beta_1$, and denoised heatmaps resulting from applying translations.

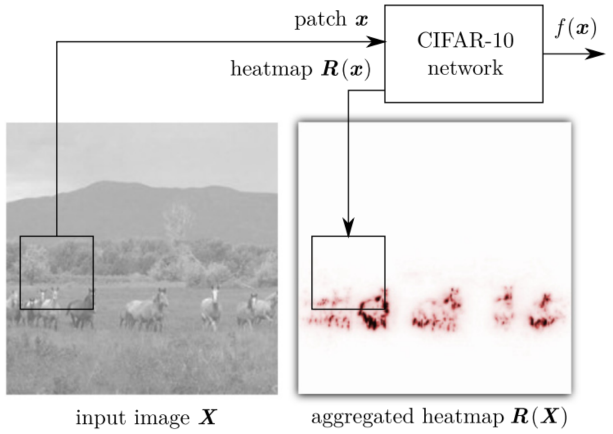
Sliding window explanations for large images

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

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 th

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

Pixels then receive relevance from all patches to which they belong and in which they contribute to the function value f (x).


patch $\boldsymbol{x}$
heatmap $\boldsymbol{R}(\boldsymbol{x})$
CIFAR-10 network
$f(\boldsymbol{x})$
input image $\boldsymbol{X}$
aggregated heatmap $\boldsymbol{R}(\boldsymbol{X})$

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.

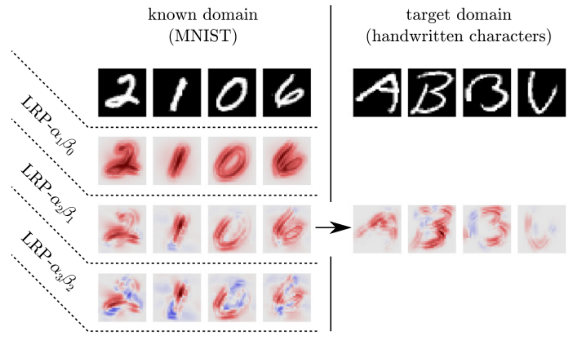
Evaluating explanation quality

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.

7.1. Transfer with a simple task.

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.

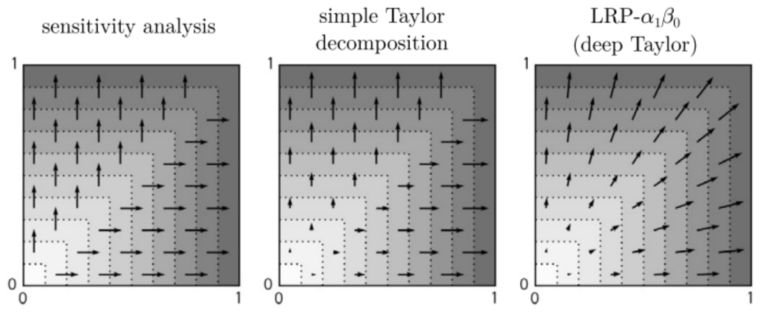


We can observe that the heatmaps look similar across domains, with a similar placement of relevance around the handwriting strokes and a similar share of negative relevance.
However, when a simple related task is not available, it becomes essential to be able to define at a more abstract level what are the characteristics of a good explanation, and to be able to test for these characteristi

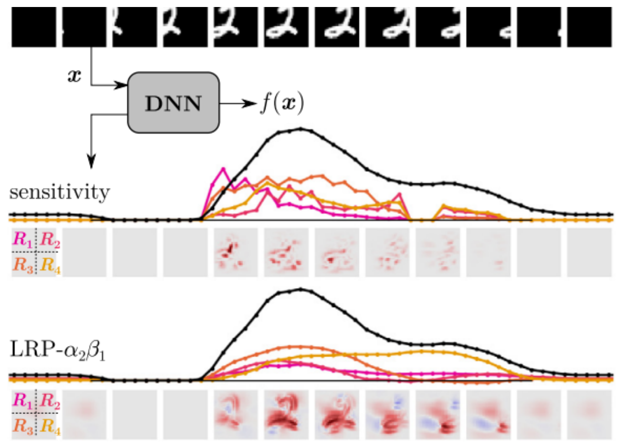
7.2. Explanation continuity

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

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



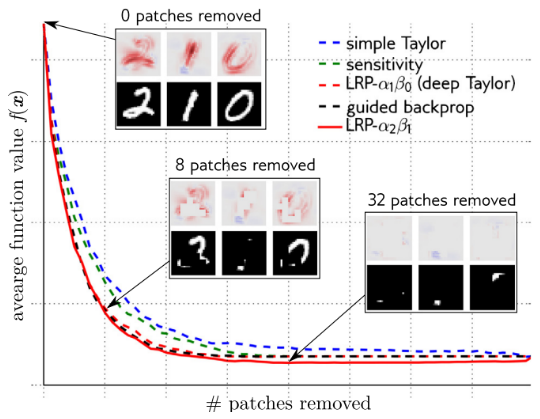
Techniques that rely on the function’s gradient, such as sensitivity analysis or simple Taylor decomposition, are also strongly exposed to derivative noise [61] that characterizes complex machine learning models.
More specifically, they are subject to the problem of shattered gradients [7,49] occurring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



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 (x) is continuous, relevance scores produced by sensitivity analysis are strongly varying.

Explanation selectivity

The method works as follows: A sharp drop of the function's value, summarized by a low AUC score, indicates that the correct features have been identified as relevant.



The choice of a patch instead of a pixel as a unit of removal lets the analysis focus on removing the actual content of the image and avoids introducing pixel artifacts.
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.

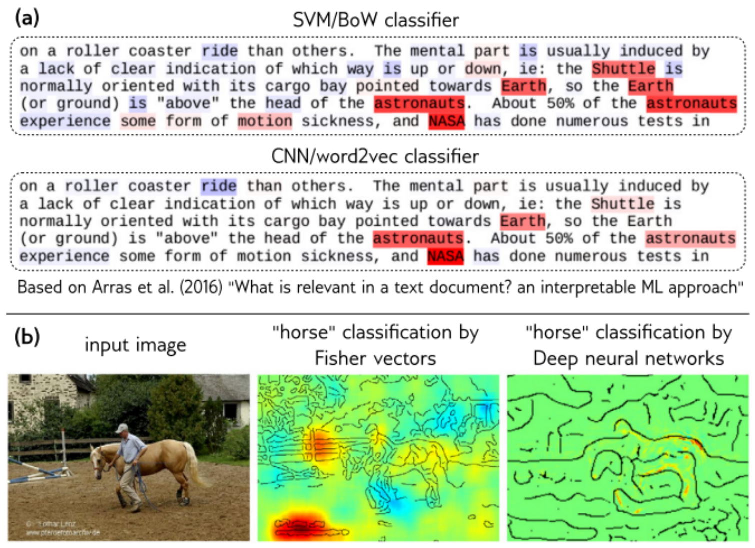
Applications

Model validation

We focus in this section on two types of applications: validation of a trained model, and analysis of scientific data.

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)
SVM/BoW classifier
CNN/word2vec classifier
(b)
input image
"horse" classification by Fisher vectors
"horse" classification by Deep neural networks

```
on a roller coaster ride than others.  The mental part is usually induced by
a lack of clear indication of which way is up or down, ie: the Shuttle is
normally oriented with its cargo bay pointed towards Earth, so the Earth
(or ground) is "above" the head of the astronauts.  About 50% of the astronauts
experience some form of motion sickness, and NASA has done numerous tests in
```

```
on a roller coaster ride than others.  The mental part is usually induced by
a lack of clear indication of which way is up or down, ie: the Shuttle is
normally oriented with its cargo bay pointed towards Earth, so the Earth
(or ground) is "above" the head of the astronauts.  About 50% of the astronauts
experience some form of motion sickness, and NASA has done numerous tests in
```

Based on Arras et al. (2016) "What is relevant in a text document? an interpretable ML approach"

Based on Lapuschkin et al. (2016) "Analyzing classifiers: Fisher vectors and deep neural nets"

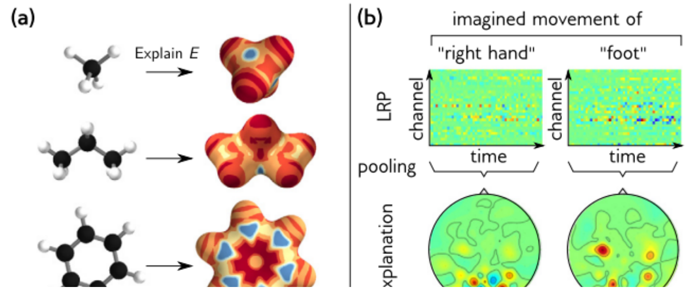
Analysis of scientific data

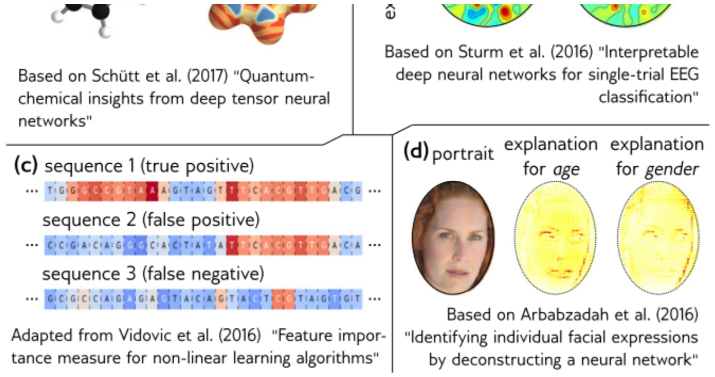
For a long time, the computational scientist would face a tradeoff between interpretability and predictive power, where linear models would sometimes be preferred to nonlinear models despite their typically lower pre

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.




Based on Schütt et al. (2017) "Quantum-chemical insights from deep tensor neural networks"
Based on Sturm et al. (2016) "Interpretable deep neural networks for single-trial EEG classification"
(c) sequence 1 (true positive)
sequence 2 (false positive)
sequence 3 (false negative)
Adapted from Vidovic et al. (2016) "Feature importance measure for non-linear learning algorithms"
(d) portrait
explanation for *age*
explanation for *gender*
Based on Arbabzadah et al. (2016) "Identifying individual facial expressions by deconstructing a neural network"

Conclusion

Building transparent machine learning systems is a convergent approach to both extracting novel domain knowledge and performing model validation.

This tutorial has covered two key directions for improving machine learning transparency:
interpreting the concepts learned by a model by building prototypes
explaining the model’s decisions by identifying the relevant input variables.

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.

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 le