# Tutorial on Interpreting and Explaining Deep Models in Computer Vision


Wojciech Samek
(Fraunhofer HHI)

Grégoire Montavon
(TU Berlin)

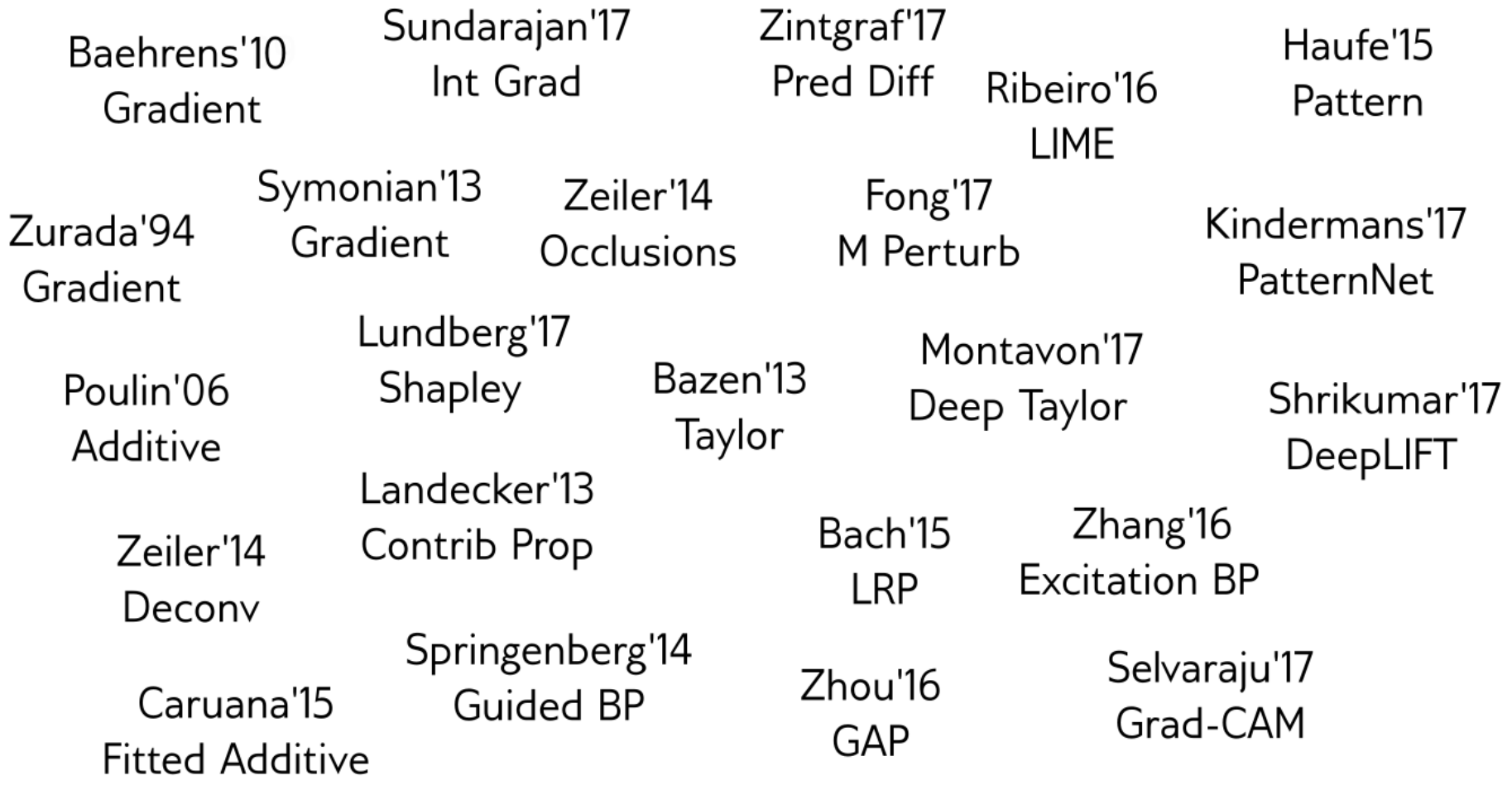
Klaus-Robert Müller
(TU Berlin)


| | |
|---|---|
| 08:30 - 09:15 | Introduction KRM |
| 09:15 - 10:00 | Techniques for Interpretability GM |
| 10:00 - 10:30 | Coffee Break ALL |
| 10:30 - 11:15 | Applications of Interpretability WS |
| 11:15 - 12:00 | Further Applications and Wrap-Up KRM |

CVPR 2018
SALT LAKE CITY • JUNE 18-22

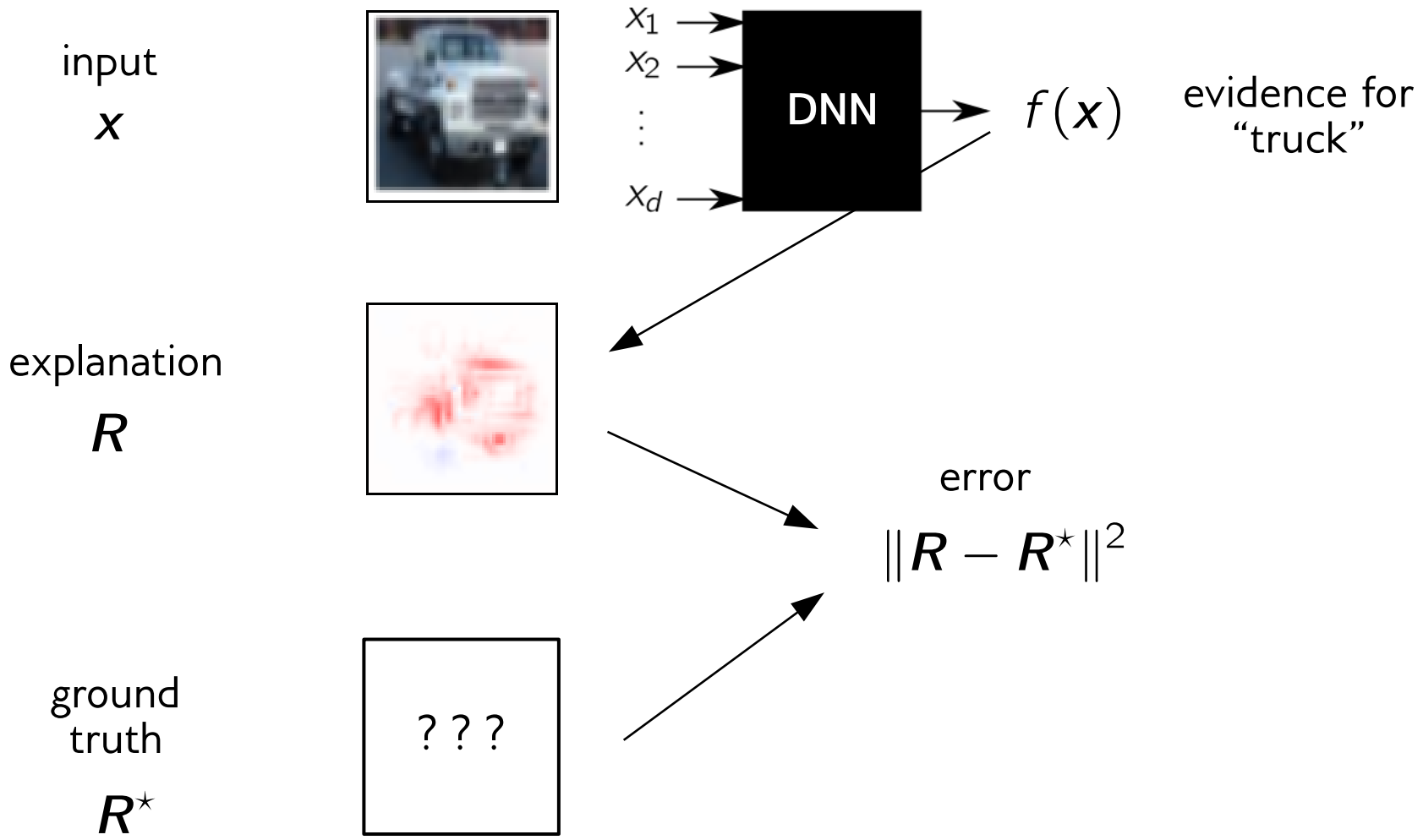
BERLIN BIG DATA CENTER

# Overview of Explanation Methods

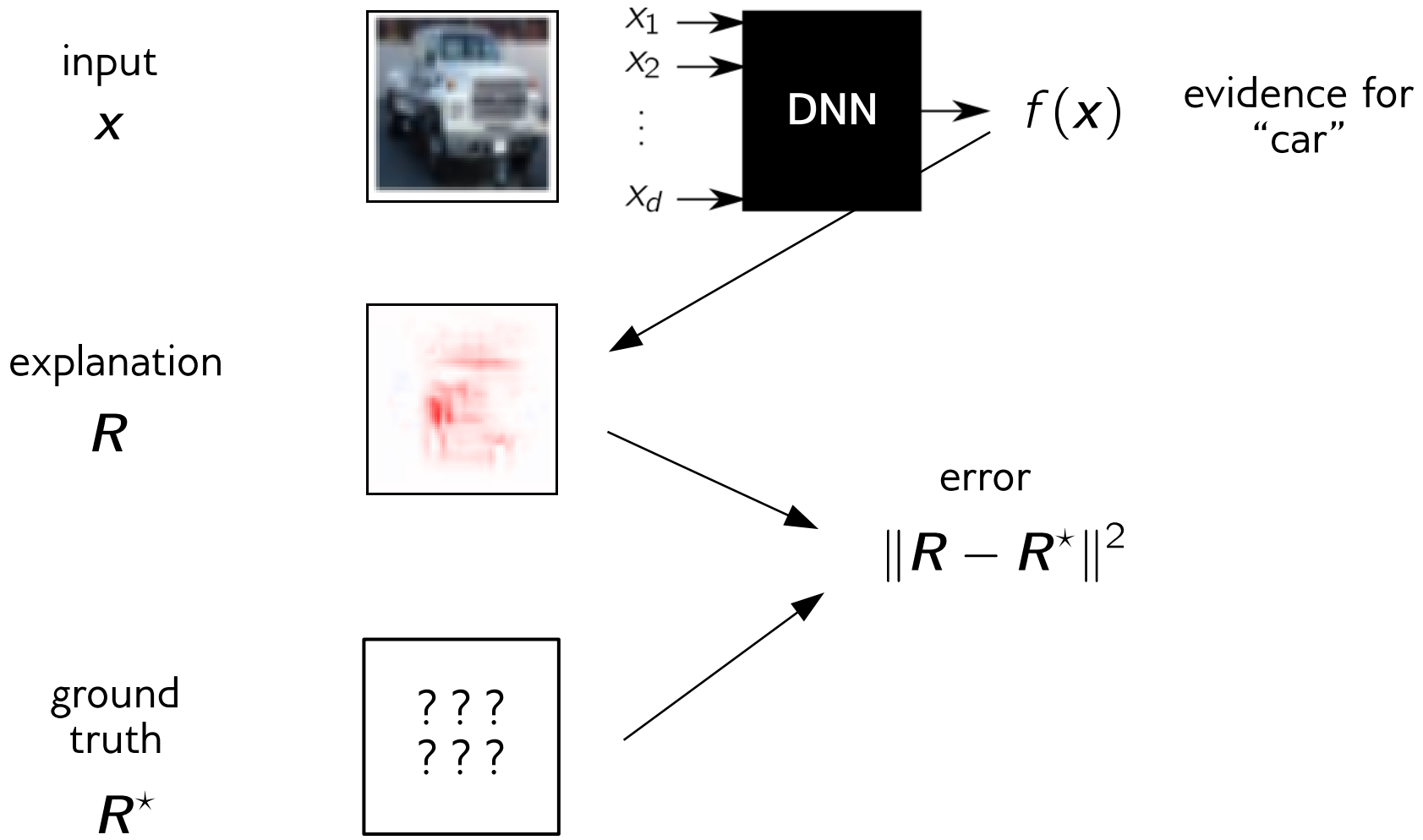



**Question:** Which one to choose ?

# First Attempt: Distance to Ground Truth



# First Attempt: Distance to Ground Truth

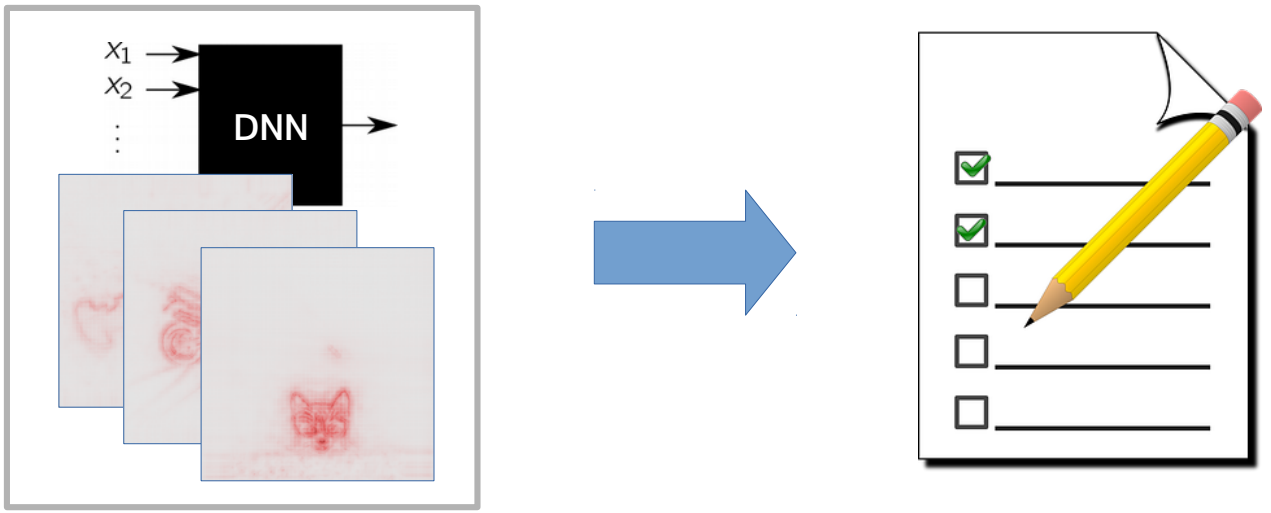


# From Ground Truth Explanations to Axioms

**Idea:** Evaluate the explanation technique axiomatically, i.e. it must pass a number of predefined "unit tests".

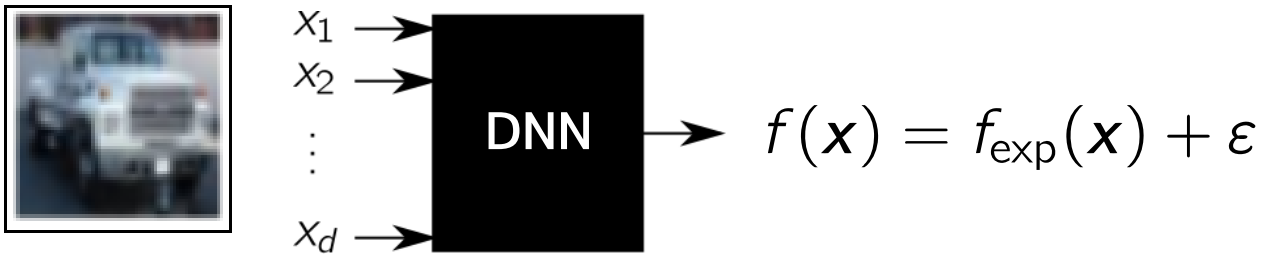
[Sun'11, Bach'15, Montavon'17, Samek'17, Sundarajan'17, Kindermans'17, Montavon'18].

explanation technique



# Properties 1-2: Conservation and Positivity

[Montavon'17, see also Sun'11, Landecker'13, Bach'15]




**Conservation:** Total attribution on the input features should be proportional to the amount of (explainable) evidence at the output.

$$\sum_{p=1}^{d} R_p = f_{\text{exp}}(\boldsymbol{x})$$

**Positivity:** If the neural network is certain about its prediction, input features are either relevant (positive) or irrelevant (zero).

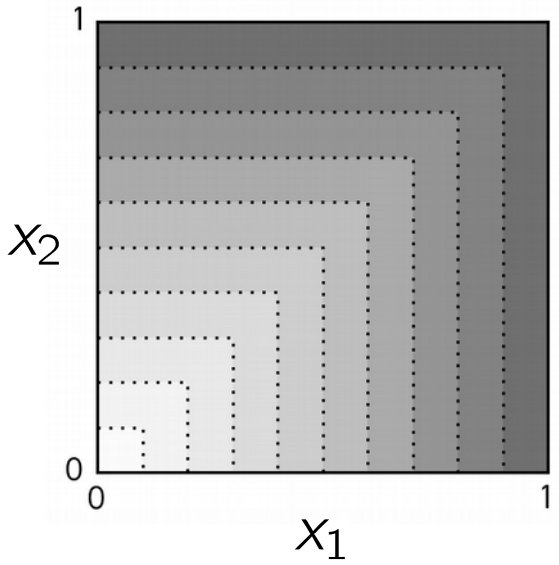
$$\forall_{p=1}^{d} : \; R_p \geq 0$$

# Property 3: Continuity [Montavon'18]

If two inputs are the almost the same, and the prediction is also almost the same, then the explanation should also be almost the same.

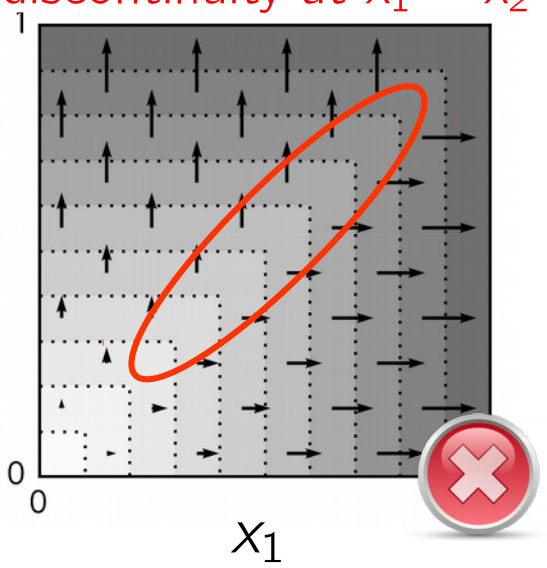
**Example:**

$f(\boldsymbol{x}) = \max(x_1, x_2)$

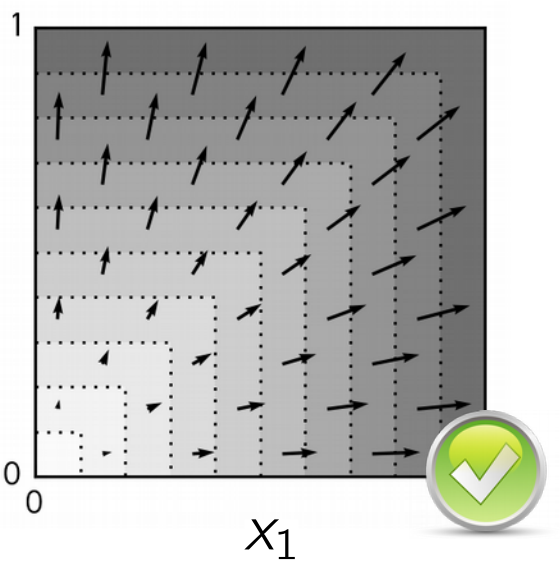



Method 1

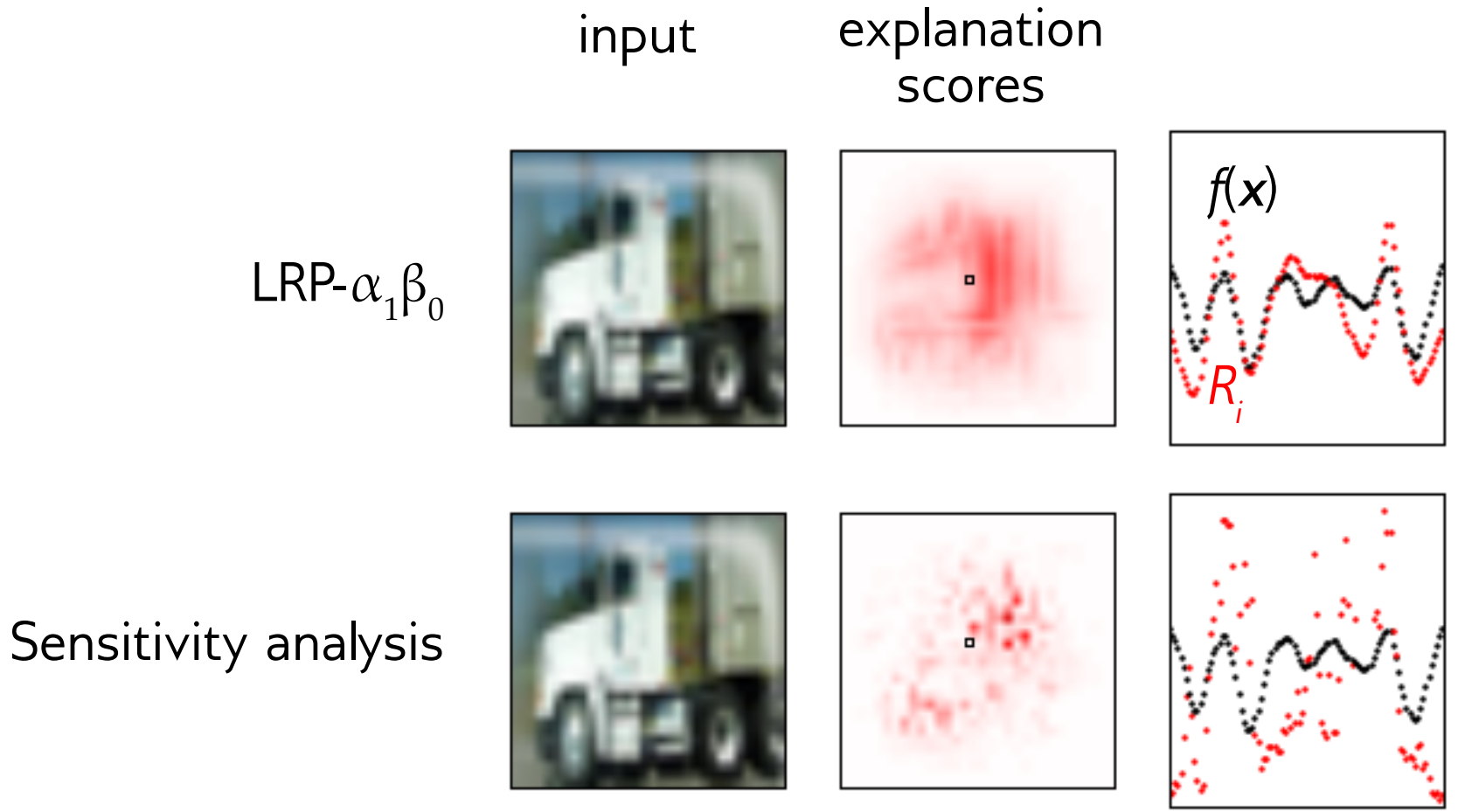
discontinuity at $x_1 = x_2$




Method 2



# Testing Continuity

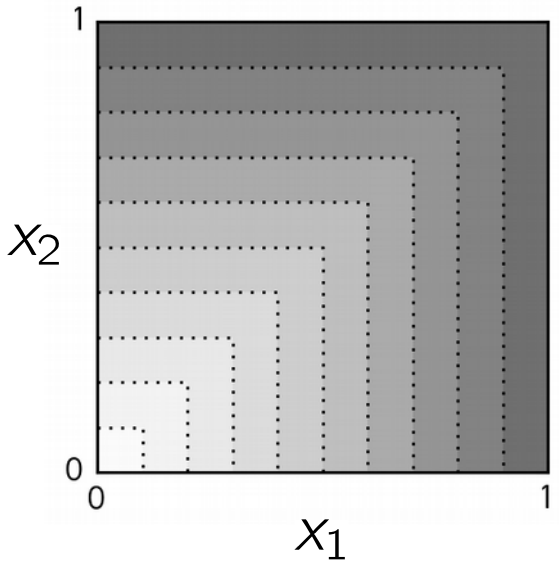


# Property 4: Selectivity [Bach'15, Samek'17]

Model must agree with the explanation: If input features are attributed relevance, removing them should reduce evidence at the output.

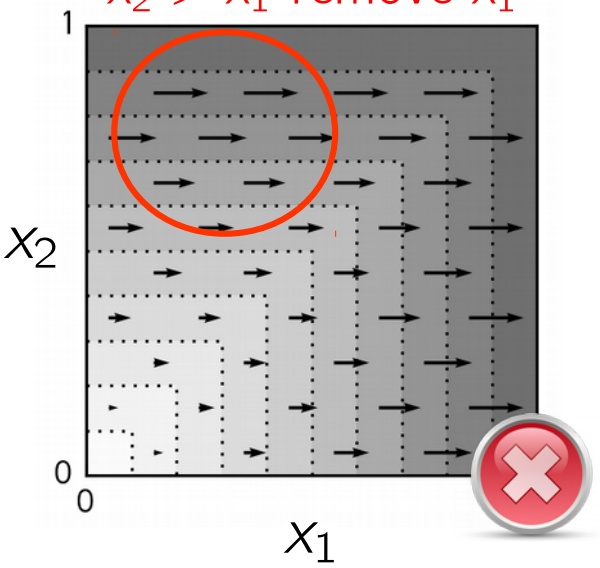
**Example:**

$f(\boldsymbol{x}) = \max(x_1, x_2)$

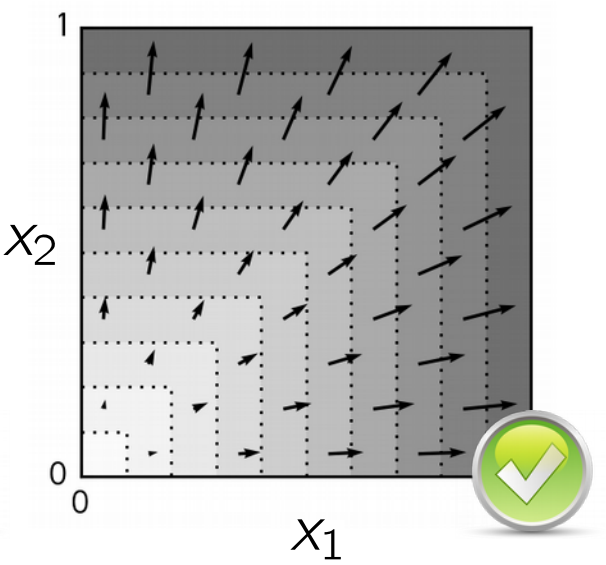



Method 1

$x_2 > x_1$ remove $x_1$

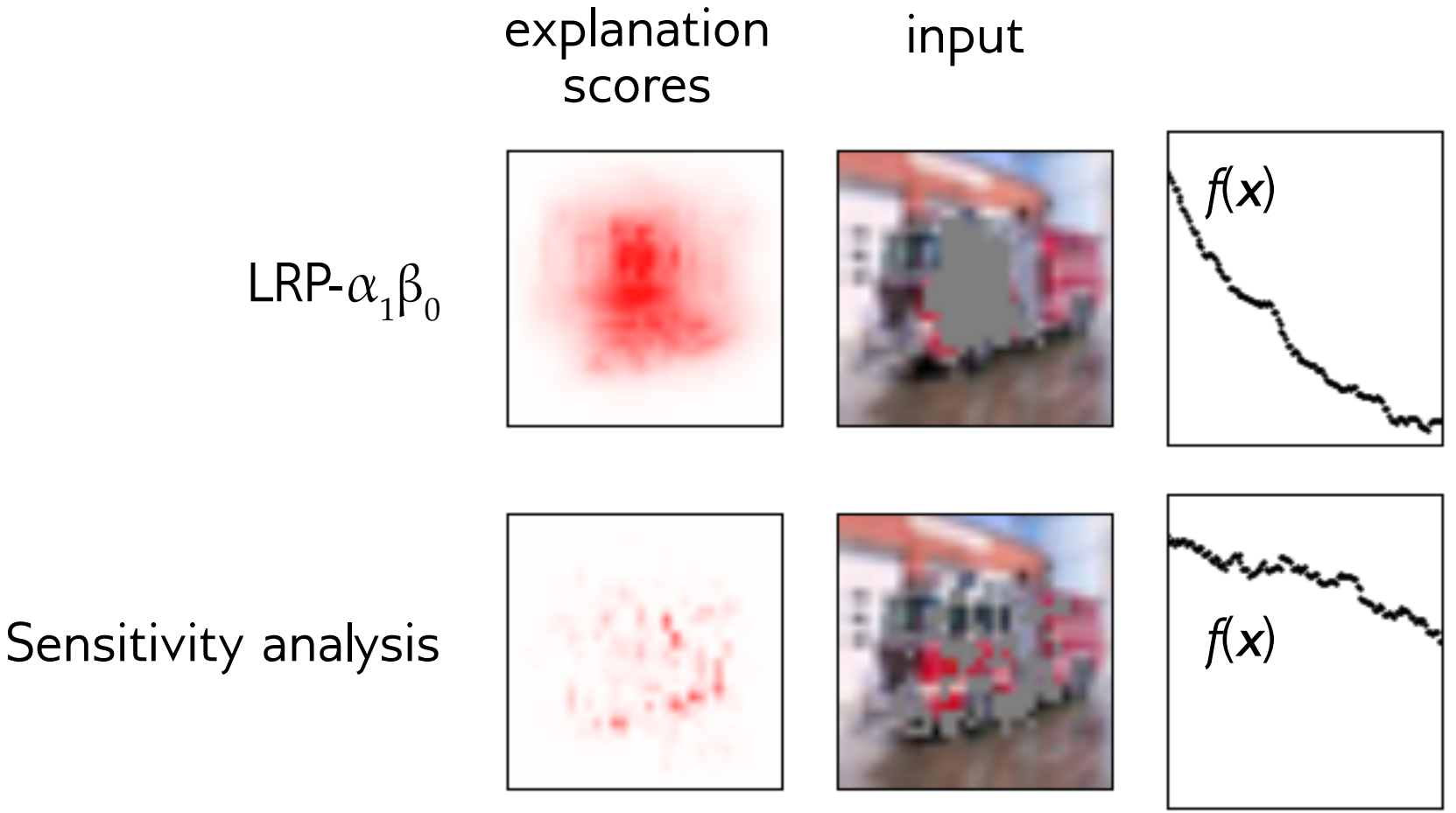



Method 2

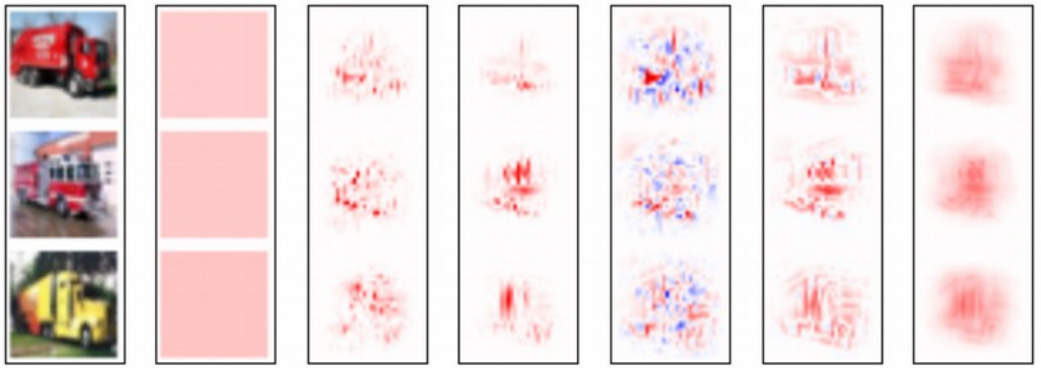


# Testing Selectivity with Pixel-Flipping

[Bach'15, Samek'17]



## Explanation techniques




## Properties

| | Uniform | (Gradient)$^2$ | (Guided BP)$^2$ | Gradient x Input | Guided BP x Input | LRP-$\alpha_1\beta_0$ | ... |
|---|---|---|---|---|---|---|---|
| 1. Conservation | ✔ | | | ✔ | ✔ | ✔ | |
| 2. Positivity | ✔ | ✔ | ✔ | | ✔ | ✔ | |
| 3. Continuity | ✔ | | ✔ | | ✔ | ✔ | |
| 4. Selectivity | | ✔ | ✔ | ✔ | ✔ | ✔ | |
| ... | | | | | | | |

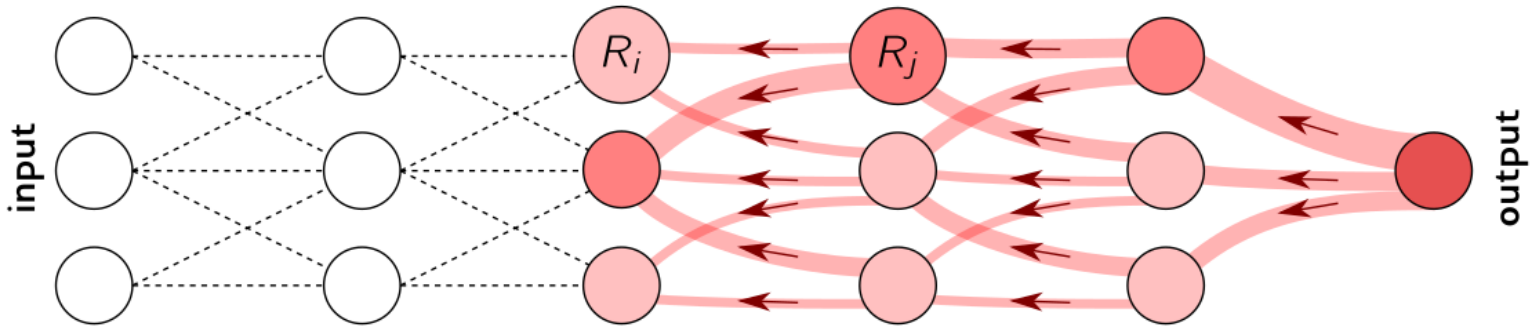
**Question:** Can we deduce some properties without experiments, directly from the equations?

# Reminder

Backprop internals (for propagating gradient)

$$a_j = \max\left(0, \sum_i a_i w_{ij} + b_j\right) \qquad \delta_i = \sum_j w_{ij} \cdot 1_{z_j>0} \cdot \delta_j$$

LRP-$\alpha_1\beta_0$ internals (for propagating relevance)




$$a_j = \max\left(0, \sum_i a_i w_{ij} + b_j\right) \qquad R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

# Example: Deducing Conservation

**LRP-$\alpha_1\beta_0$ propagation rule**

$$R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

Summing gives the property

$$\sum_i R_i = \sum_j \frac{\sum_i a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

vs. grad × input

$$\delta_i = \sum_j w_{ij} 1_{z_j>0} \delta_j$$

× input

$$\sum_i a_i \delta_i = \sum_j \frac{\sum_i a_i w_{ij}}{\sum_i a_i w_{ij} + b_j} a_j \delta_j$$

When bias is negative, grad × input will tend to inflate scores.

$$\sum_{p=1}^d R_p = \cdots = \sum_i R_i = \sum_j R_j = \ldots = f(\boldsymbol{x})$$

# Example: Deducing Continuity

**LRP-$\alpha_1\beta_0$ propagation rule**

$$R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

$$\Rightarrow \quad \underbrace{a_i c_i}_{R_i} = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} \underbrace{a_j c_j}_{R_j}$$

$$\Rightarrow \quad c_i = \sum_j w_{ij}^+ \frac{\left(\sum_i a_i w_{ij} + b_j\right)^+}{\sum_i a_i w_{ij}^+} c_j$$

(when bias negative, continuity due to denominator upper-bounding numerator.)

vs. grad × input

$$\delta_i = \sum_j w_{ij} \cdot 1_{z_j>0} \cdot \delta_j$$

$\cdots \Leftarrow c_i(\boldsymbol{a}, (c_j)_j)$ continuous $\Leftarrow \cdots \Leftarrow 1$ continuous

# Intermediate Conclusion

Ground-truth explanations are elusive. In practice, we are reduced to visual assessment or to test the explanation for a number of axioms.

Some properties can be deduced from the structure of the explanation method. Other can be tested empirically.

LRP-$\alpha_1\beta_0$ satisfies key properties of an explanation. Sensitivity analysis and gradient × input have crucial limitations.

# From LRP to Deep Taylor Decomposition

The LRP-$\alpha_1\beta_0$ rule

$$R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

can be seen as

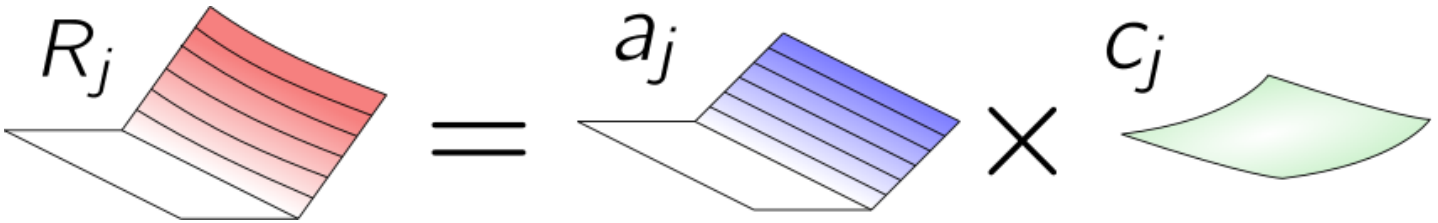
a deep Taylor decomposition (DTD) [Montavon'17]

which then yields

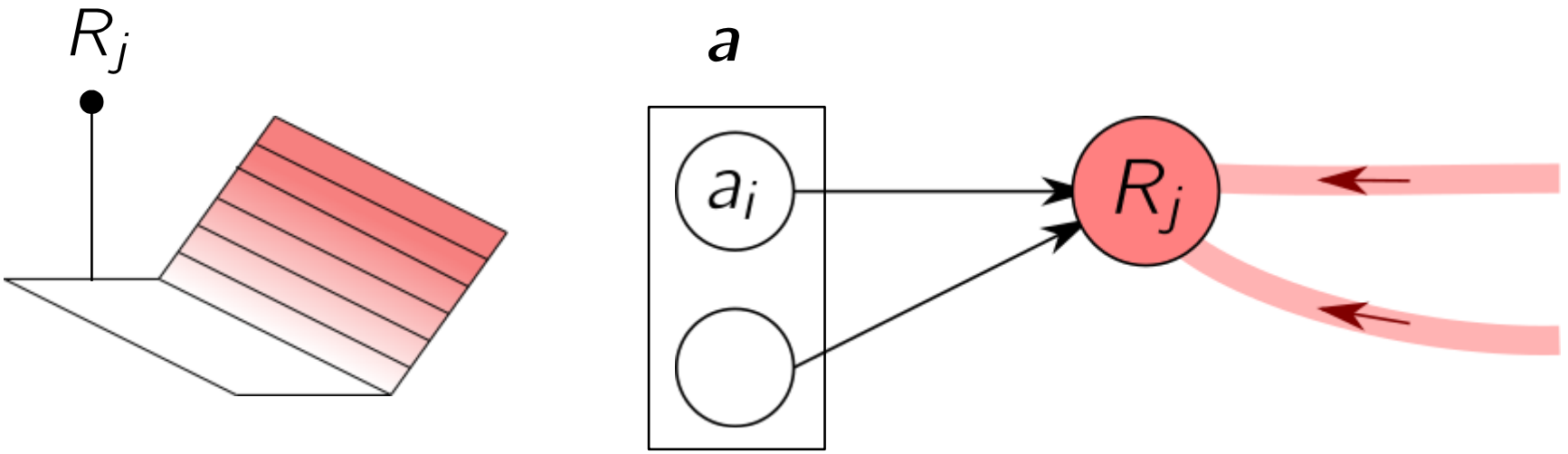
domain- and layer-specific rules

# DTD: The Structure of Relevance

**Proposition:** Relevance at each layer is a product of the activation and an approximately constant term.

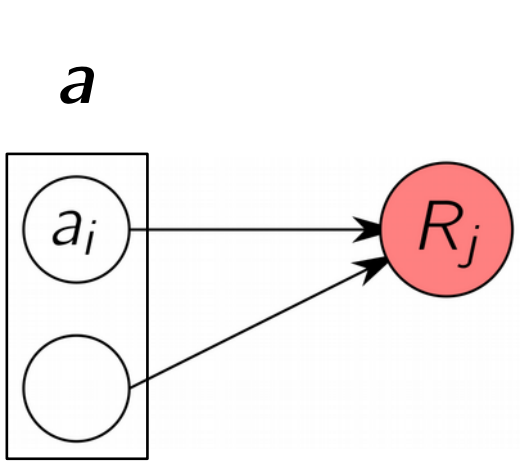


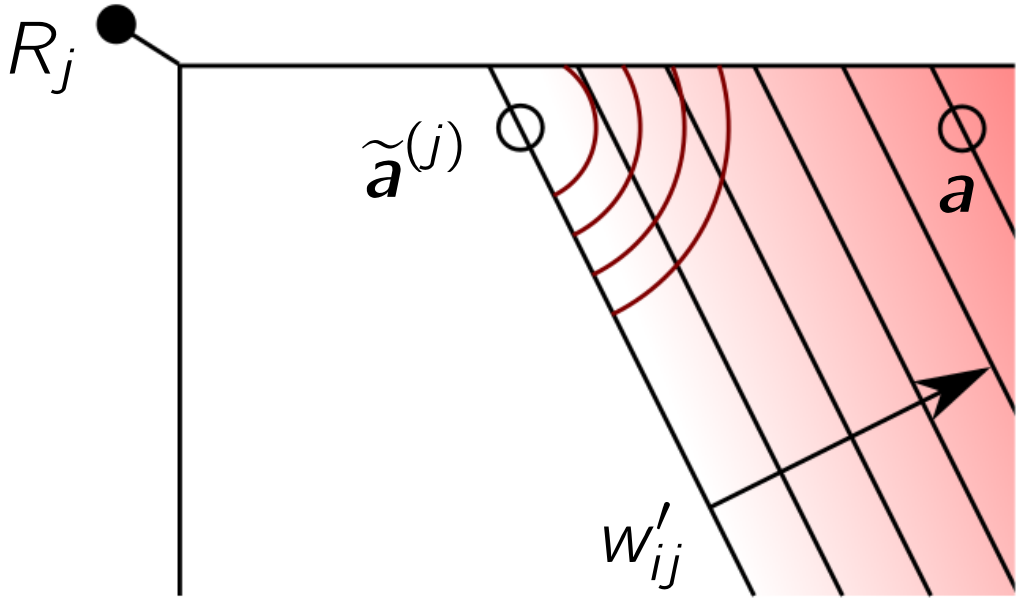
# DTD: The Relevance as a Neuron




$$R_j(\boldsymbol{a}) = \max(0, \sum_i a_i w_{ij} + b_j) \cdot c_j$$
$$= \max(0, \sum_i a_i \underbrace{w_{ij} c_j}_{w'_{ij}} + \underbrace{b_j c_j}_{b'_j})$$

# DTD: Taylor Expansion of the Relevance




$$R_j(\boldsymbol{a}) = R_j(\widetilde{\boldsymbol{a}}^{(j)}) + \sum_i \frac{\partial R_j}{\partial a_i}\Big|_{\widetilde{\boldsymbol{a}}^{(j)}} \cdot (a_i - \widetilde{a}_i^{(j)}) + \varepsilon$$

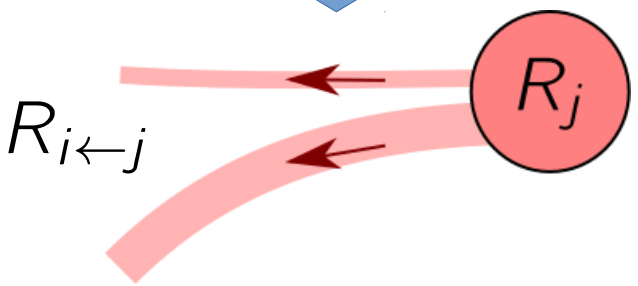
# DTD: Decomposing the Relevance

Taylor expansion at root point:

$$R_j(\boldsymbol{a}) = R_j(\widetilde{\boldsymbol{a}}^{(j)}) + \sum_i \frac{\partial R_j}{\partial a_i}\Big|_{\widetilde{\boldsymbol{a}}^{(j)}} \cdot (a_i - \widetilde{a}_i^{(j)}) + \varepsilon$$

$$0 \qquad \frac{(a_i - \widetilde{a}_i^{(j)})w_{ij}}{\sum_i (a_i - \widetilde{a}_i^{(j)})w_{ij}} R_j \qquad 0$$

Relevance can now be backward propagated



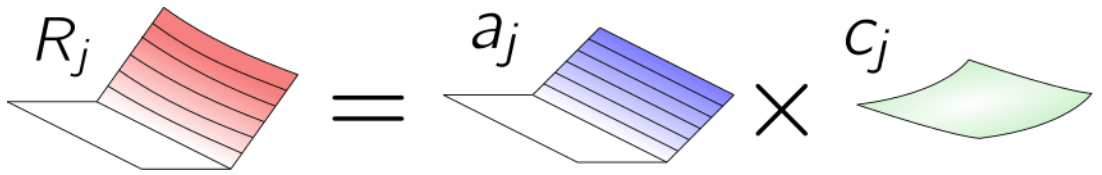
# DTD: Choosing the Root Point

$$R_{i \leftarrow j} = \frac{(a_i - \widetilde{a}_i^{(j)}) w_{ij}}{\sum_i (a_i - \widetilde{a}_i^{(j)}) w_{ij}} R_j \qquad \text{(Deep Taylor generic)}$$

| **Choice of root point** | | $\widetilde{\boldsymbol{a}}^{(j)} \in \mathcal{D}$ | $\|\boldsymbol{a} - \widetilde{\boldsymbol{a}}^{(j)}\|$ |
|---|---|---|---|
| 1. nearest root | $\widetilde{\boldsymbol{a}}^{(j)} = \boldsymbol{a} - t \cdot \boldsymbol{w}_j$ | | ✔ |
| 2. rescaled activation | $\widetilde{\boldsymbol{a}}^{(j)} = \boldsymbol{a} - t \cdot \boldsymbol{a}$ | ✔ | |
| 3. rescaled excitations | $\widetilde{\boldsymbol{a}}^{(j)} = \boldsymbol{a} - t \cdot \boldsymbol{a} \odot \mathbf{1}_{\boldsymbol{w}_j \succ 0}$ | ✔ | ✔ |

$$R_{i \leftarrow j} = \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j \qquad \text{(LRP-}\alpha_1\beta_0\text{)}$$

# DTD: Verifying the Product Structure




But was it true?

**1.** assume it holds in higher-layer

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} a_k c_k = a_j \underbrace{\sum_k w_{jk}^+ \frac{(\sum_j a_j w_{jk} + b_k)^+}{\sum_j a_j w_{jk}^+} c_k}_{c_j}$$

**2.** apply LRP-$\alpha_1\beta_0$ rule

**3.** observe it also holds in lower-layer

# From LRP to Deep Taylor Decomposition

The LRP-$\alpha_1\beta_0$ rule

$$R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

can be seen as

a deep Taylor decomposition (DTD) [Montavon'17]

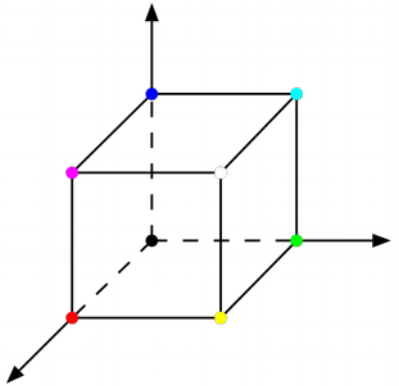
which then yields

domain- and layer-specific rules

# DTD: Application to Input Layers

**Pixels:**

$$x \in [l, h]^{3 \times d}$$



**Embeddings:**

$$x \in \mathbb{R}^d$$

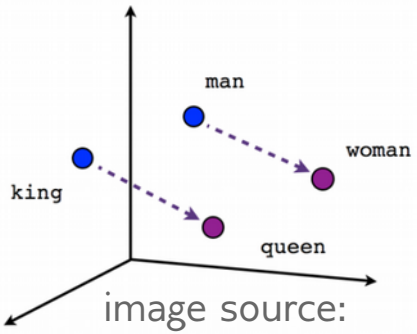



image source:
Tensorflow tutorial

1. Choose a root point that is nearby and satisfies domain constraints

$$(x - \widetilde{x}^{(j)}) = t \cdot (x - l \odot 1_{w_j \succ 0} - h \odot 1_{w_j \prec 0})$$

$$(x - x^{(j)}) = t \cdot w_j$$

2. Inject it in the generic DTD rule to get the specific rule

$$R_p = \sum_j \frac{x_{pj} w_{pj} - l_p w_{pj}^+ - h_p w_{pj}^-}{\sum_p x_{pj} w_{pj} - l_p w_{pj}^+ - h_p w_{pj}^-} R_j$$

$$R_p = \sum_j \frac{w_{pj}^2}{\sum_p w_{pj}^2} R_j$$

# DTD: Application to Pooling Layers

A sum-pooling layer over positive activations is equivalent to a ReLU layer with weights 1.

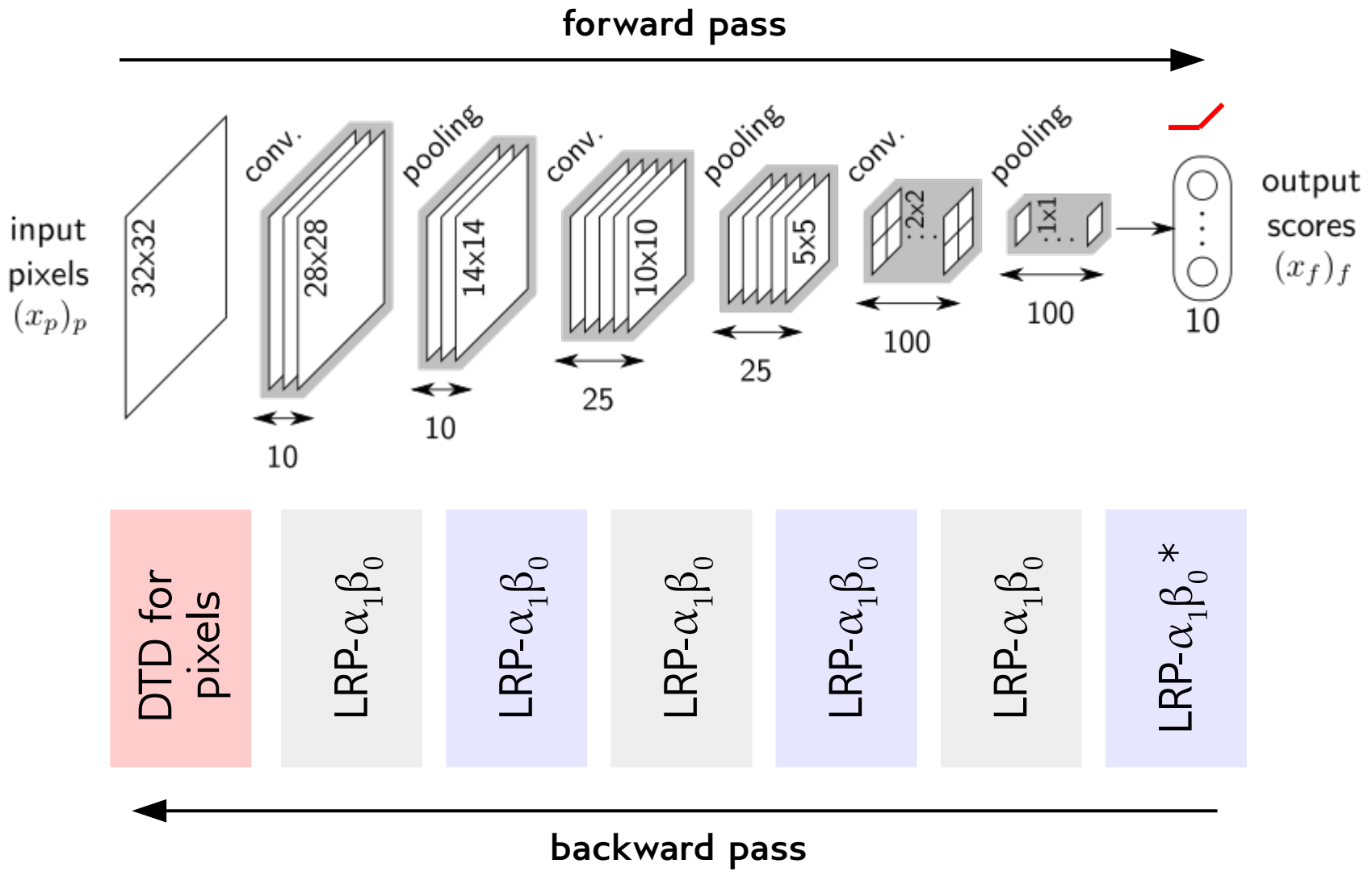
$$a_j = \left(\sum_i a_i\right) = \max\left(0, \sum_i a_i 1_{ij} + 0_j\right)$$

A $p$-norm pooling layer can be approximated as a sum-pooling layer multiplied by a ratio of norms that we treat as constant [Montavon'17].

$$a_j = \left(\sum_i a_i\right) \cdot \frac{\|(a_i)_i\|_p}{\|(a_i)_i\|_1}$$

**→ Treat pooling layers as ReLU detection layers**

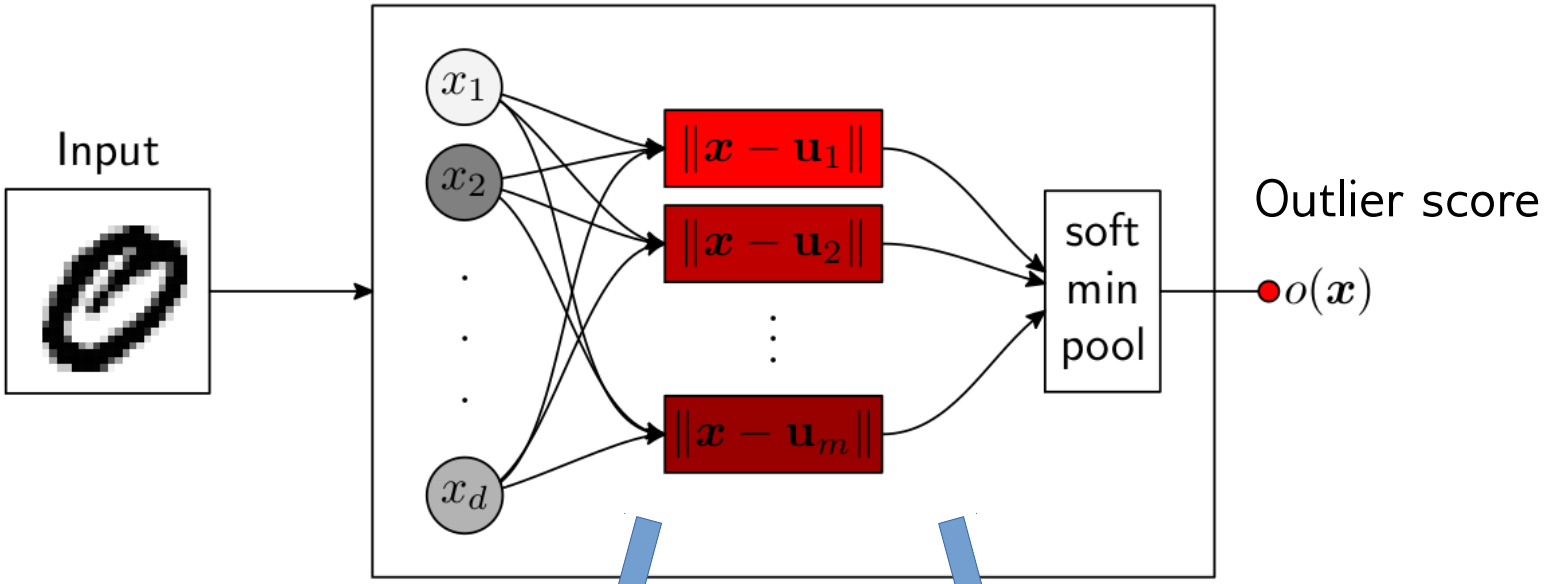
# Basic Recommendation for CNNs




* For top-layers, other rules may improve selectivity

# DTD for Kernel Models [Kauffmann'18]

1. Build a neural network equivalent of the One-Class SVM:




2. Computes its deep Taylor decomposition

$$R_i = \sum_j \frac{(x_i - u_{ij})^2}{\|\boldsymbol{x} - \boldsymbol{u}_j\|_2^2}(R_j - D_j^+)$$

**Gaussian/Laplace Kernel**

$$R_j = (a_j + \varepsilon_j) \cdot \frac{\exp(-a_j)}{\sum_j \exp(-a_j)}$$

**Student Kernel**

$$R_j = a_j \cdot \mathbb{H}[(h_{j'}/h_j)_{j'}]$$

# Implementing the LRP-$\alpha_1\beta_0$ rule

Propagation rule to implement:

$$R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

| Sequence of element-wise computations | Sequence of vector computations |
|---|---|
| $z_j \to \sum_i a_i w_{ij}^+$ | $\boldsymbol{z} \to W_+^\top \cdot \boldsymbol{a}$ |
| $s_j \to R_j / z_j$ | $\boldsymbol{s} \to \boldsymbol{R} \oslash \boldsymbol{z}$ |
| $c_i \to \sum_j w_{ij}^+ s_j$ | $\boldsymbol{c} \to W_+ \cdot \boldsymbol{s}$ |
| $R_i \to a_i c_i$ | $\boldsymbol{R} \to \boldsymbol{a} \odot \boldsymbol{c}$ |

# Implementing the LRP-$\alpha_1\beta_0$ rule

Propagation rule to implement:

$$R_i = \sum_j \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$

Code that reuses forward and gradient computations:

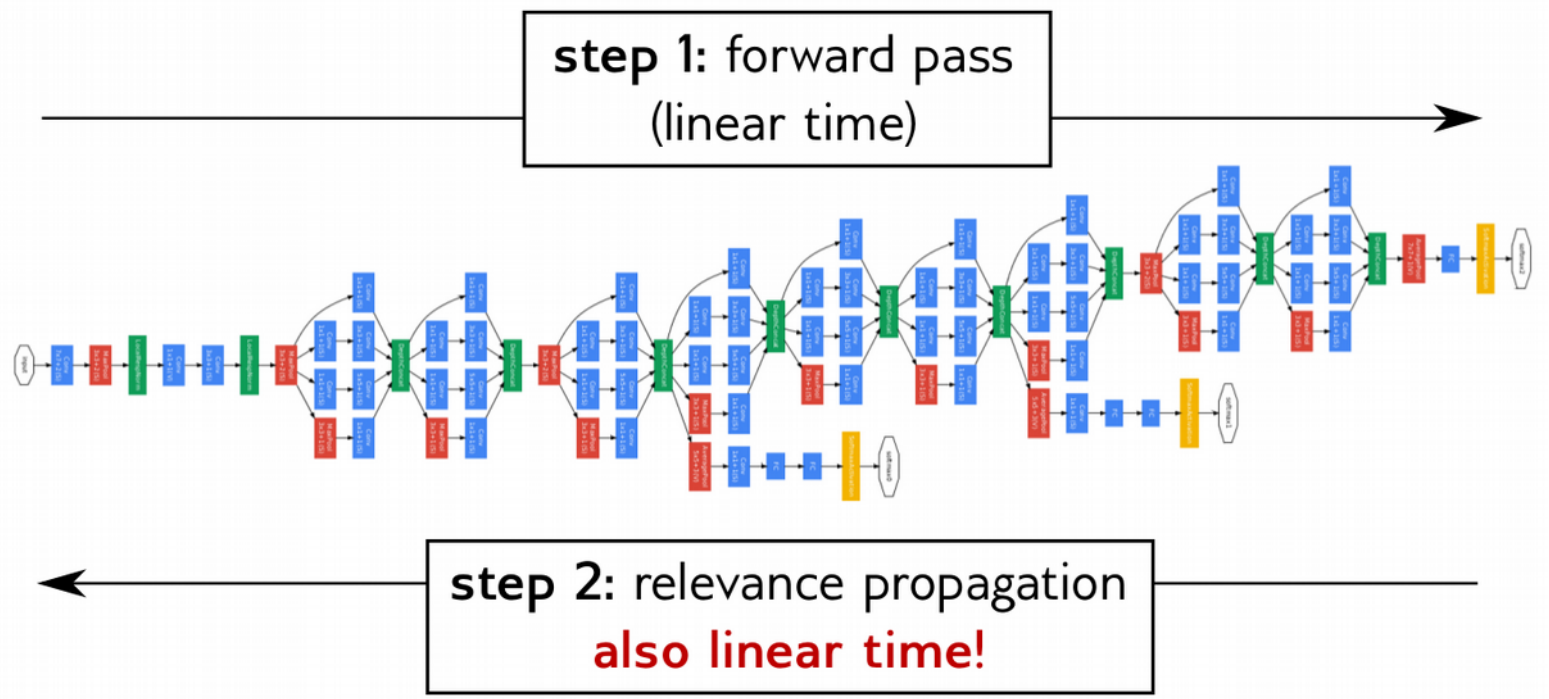
```
def lrp(layer,a,R):

    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
```

# How LRP Scales




No need for much computing power. GoogleNet explanation for single image can be done on the CPU.

Linear time scaling allows to use LRP for real-time processing, or as part of training.

# Conclusion

1. Ground-truth explanations are elusive. In practice, we are reduced to visual assessment or to test the explanation for a number of axioms.

2. Some properties can be deduced from the structure of the explanation method. Other can be tested empirically.

3. LRP-$\alpha_1\beta_0$ satisfies key properties of an explanation. Sensitivity analysis and gradient × input have crucial limitations.

4. This suitable LRP-$\alpha_1\beta_0$ propagation rule can be seen as performing a deep Taylor decomposition for deep ReLU nets.

5. The deep Taylor decomposition allows to consistently extend the framework to new models and new types of data.

# References


S Bach, A Binder, G Montavon, F Klauschen, KR Müller, W Samek. On Pixel-wise Explanations for Non-Linear Classifier Decisions by Layer-wise Relevance Propagation. PLOS ONE, 10(7):e0130140 (2015)

J Kauffmann, KR Müller, G Montavon: Towards Explaining Anomalies: A Deep Taylor Decomposition of One-Class Models. CoRR abs/1805.06230 (2018)

PJ Kindermans, S Hooker, J Adebayo, M Alber, K Schütt, S Dähne, D Erhan, B Kim: The (Un)reliability of saliency methods. CoRR abs/1711.00867 (2017)

W Landecker, M Thomure, L Bettencourt, M Mitchell, G Kenyon, S Brumby: Interpreting individual classifications of hierarchical networks. CIDM 2013: 32-38

G Montavon, S Lapuschkin, A Binder, W Samek, KR Müller: Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognition 65: 211-222 (2017)

G Montavon, W Samek, KR Müller: Methods for interpreting and understanding deep neural networks. Digital Signal Processing 73: 1-15 (2018)

W Samek, A Binder, G Montavon, S Lapuschkin, KR Müller: Evaluating the Visualization of What a Deep Neural Network Has Learned. IEEE Trans. Neural Netw. Learning Syst. 28(11): 2660-2673 (2017)

Y Sun, M Sundararajan. Axiomatic attribution for multilinear functions. EC 2011: 177-178

M Sundararajan, A Taly, Q Yan: Axiomatic Attribution for Deep Networks. ICML 2017: 3319-3328