Tutorials

Interpretable Deep Learning: Towards Understanding & Explaining DNNs

Part 2: Methods of Explanation

Wojciech Samek, Grégoire Montavon, Klaus-Robert Müller

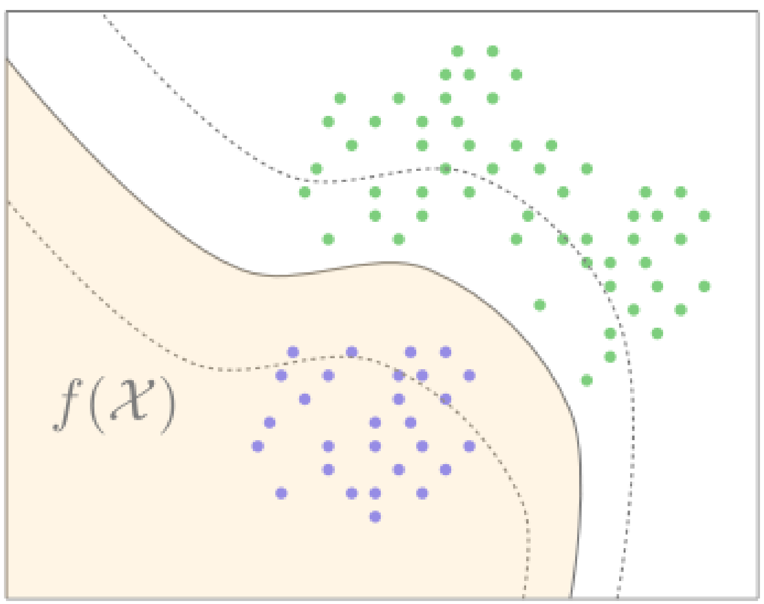
Technische Universität Berlin

Fraunhofer Heinrich Hertz Institute

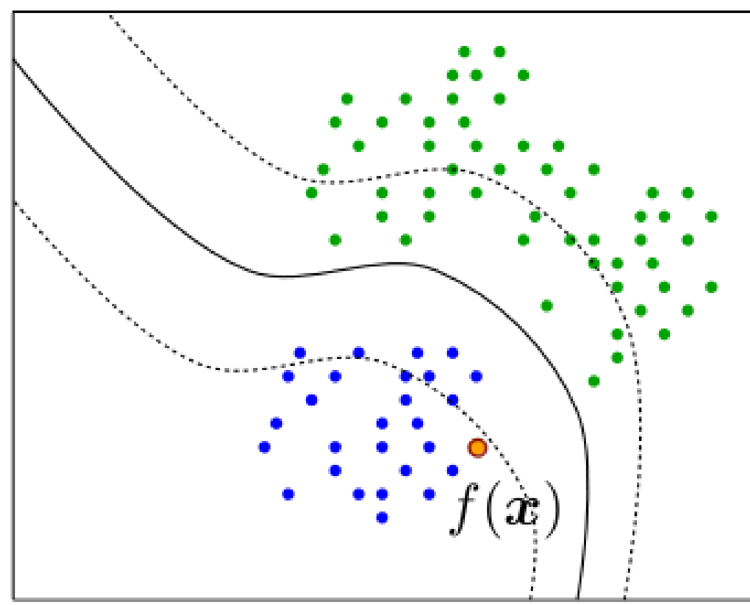
Berliner Zentrum für MASCHINELLES LERNEN

What Will be Covered in Part 2

interpreting predicted classes

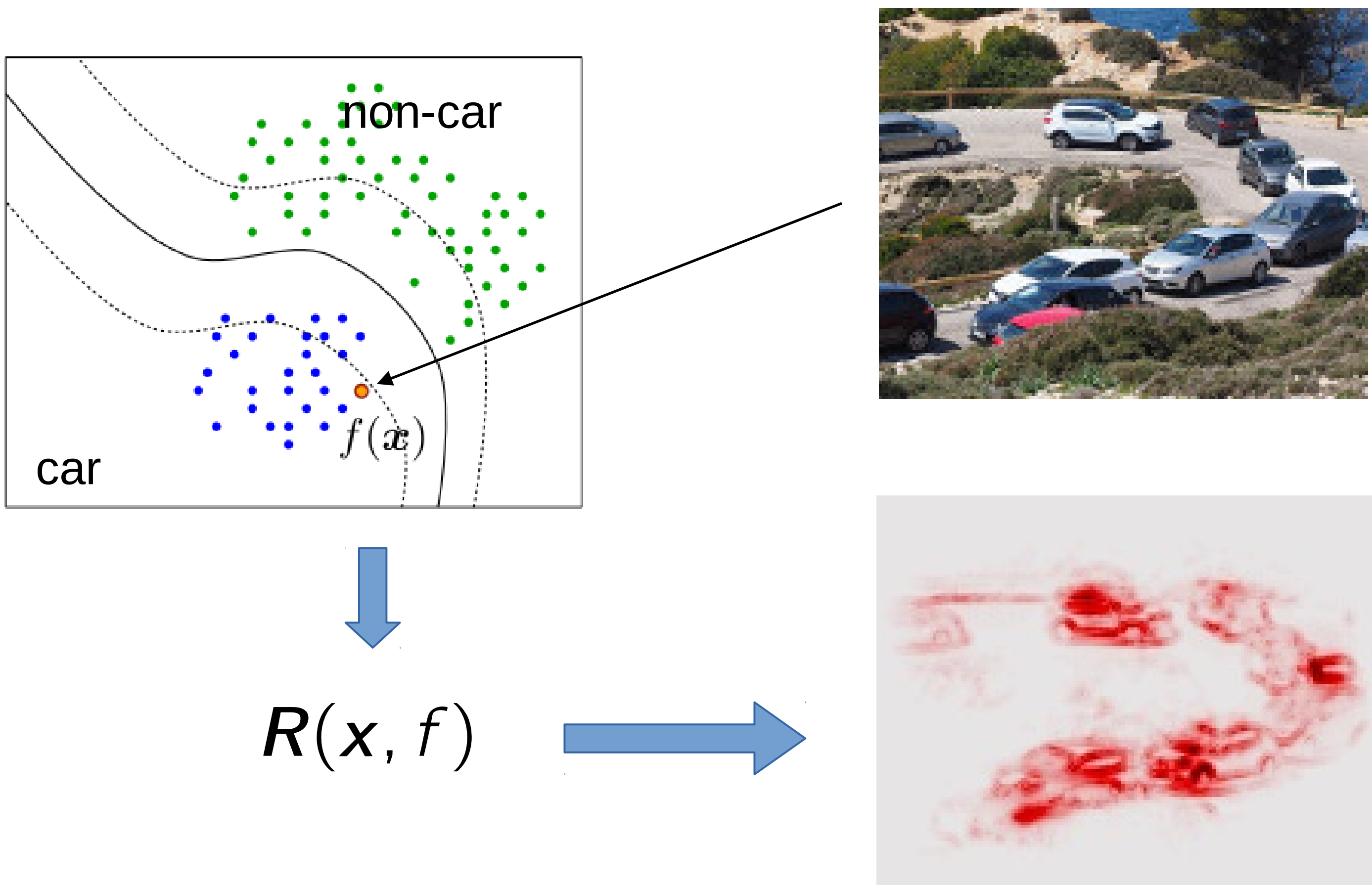


explaining individual decisions



Explaining Individual Decisions

Q: Where in the image the neural networks sees evidence for a car?



Examples of Methods that Explain Decisions

Baehrens'10
Gradient

Sundararajan'17
Int Grad

Zintgraf'17
Pred Diff

Ribeiro'16
LIME

Haufe'15
Pattern

Zurada'94
Gradient

Symonian'13
Gradient

Zeiler'14
Occlusions

Fong'17
M Perturb

Kindermans'17
PatternNet

Poulin'06
Additive

Lundberg'17
Shapley

Bazen'13
Taylor

Montavon'17
Deep Taylor

Shrikumar'17
DeepLIFT

Zeiler'14
Deconv

Landecker'13
Contrib Prop

Bach'15
LRP

Zhang'16
Excitation BP

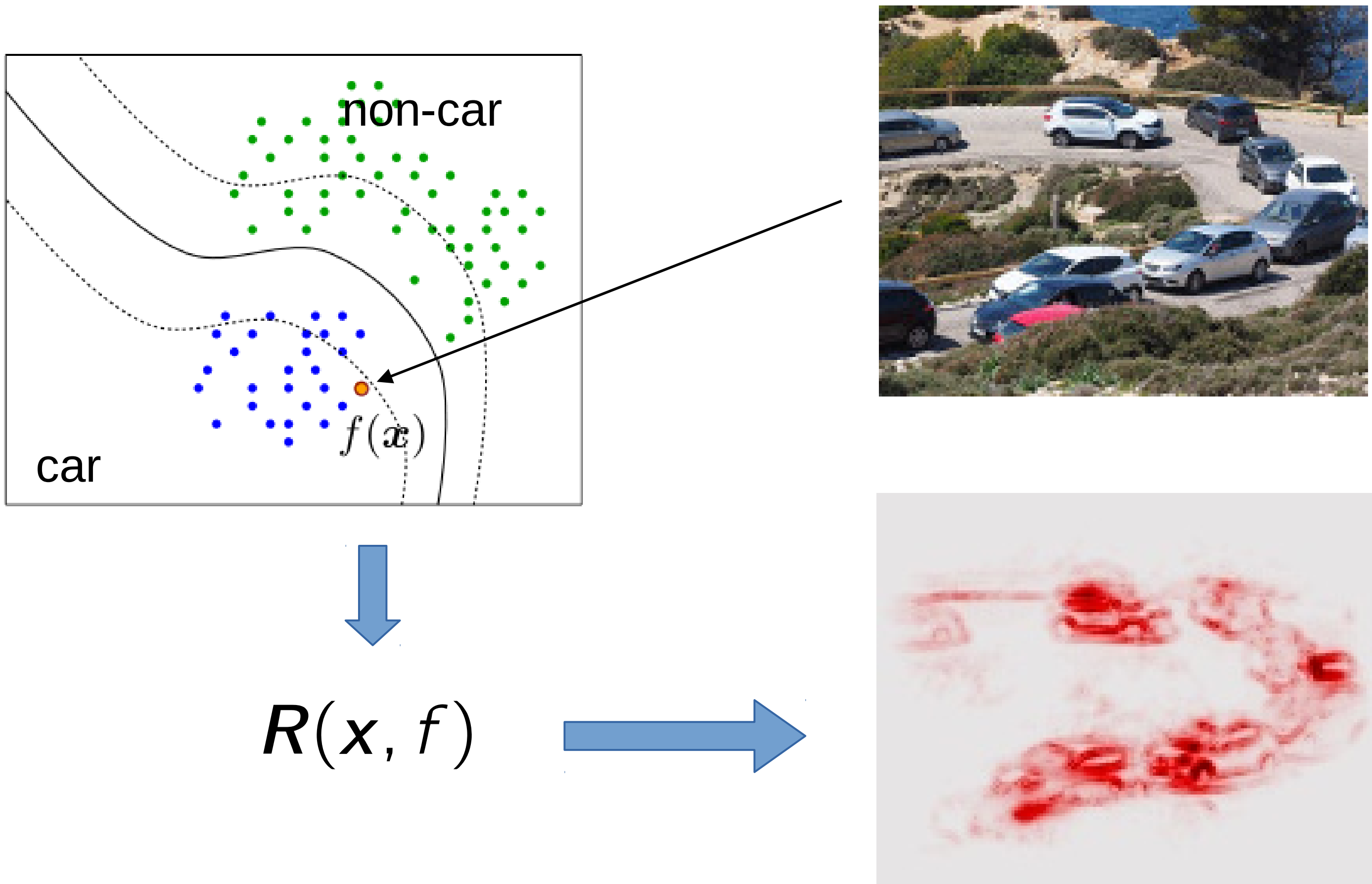
Caruana'15
Fitted Additive

Springenberg'14
Guided BP

Zhou'16
GAP

Selvaraju'17
Grad-CAM

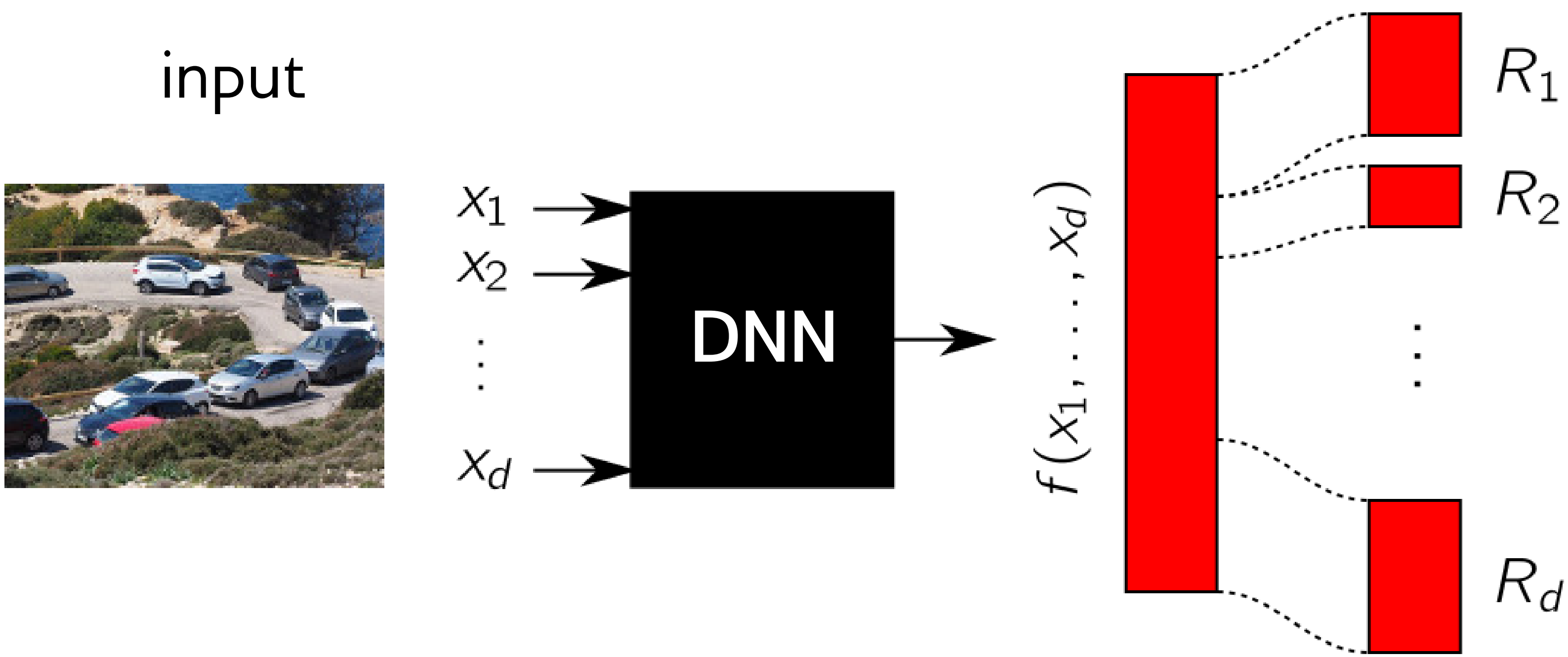
Explaining Individual Decisions



Q: In which proportion has each car contributed to the prediction?

Explaining by Decomposing

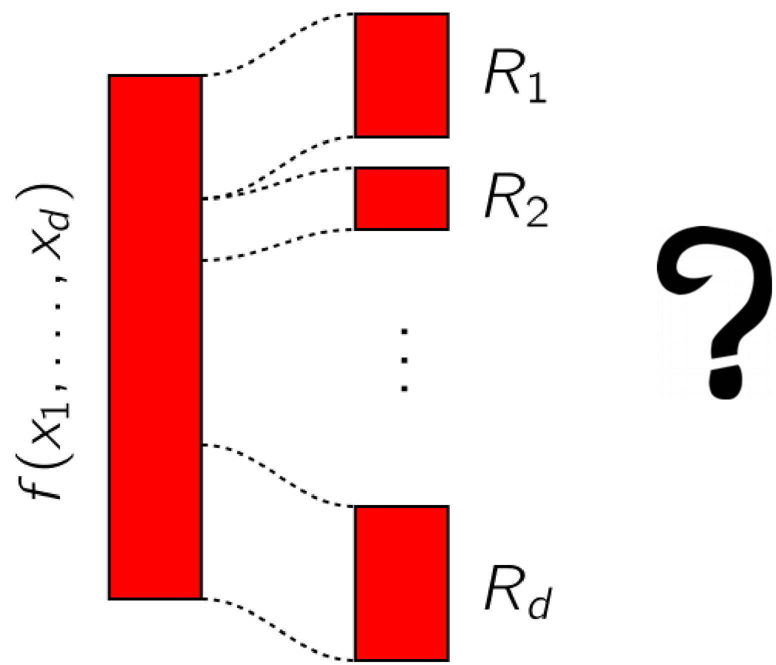
Goal: Determine the share of the output that should be attributed to each input variable.



Decomposition property: $\sum_{i=1}^{d} R_i = f(x_1, \ldots, x_d)$

Explaining by Decomposing

Goal: Determine the <u>share</u> of the output that should be attributed to each input variable.



Decomposing a prediction is generally difficult.

Sensitivity Analysis

input

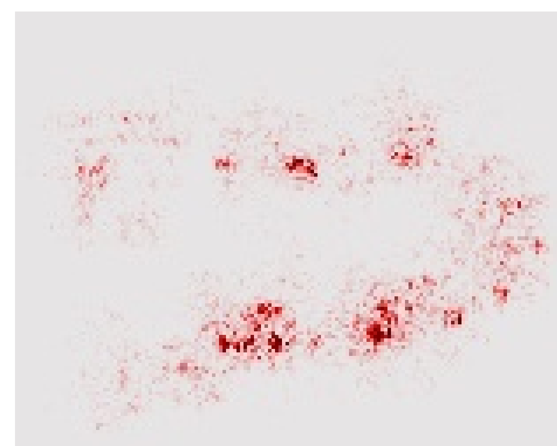
x_1, x_2, $\vdots$, x_d → **DNN** → $f(x_1, \ldots, x_d)$

evidence for "car"

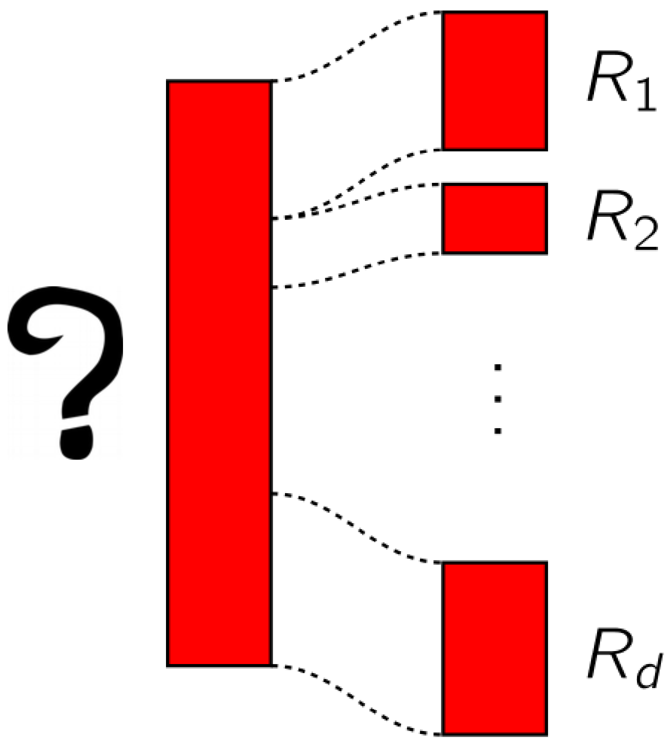
explanation for "car" (heatmap):

computes for each pixel:

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

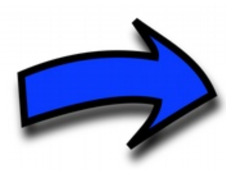


Sensitivity Analysis



Question: If sensitivity analysis computes a decomposition of something: Then, *what* does it decompose?

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



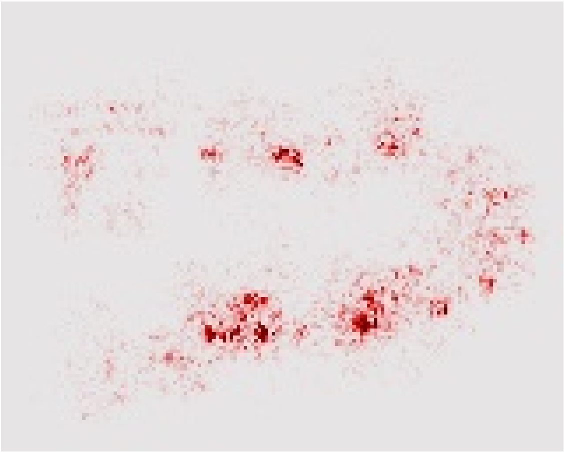
$$\sum_{i=1}^{d} R_i = \|\nabla f(\boldsymbol{x})\|^2$$

Sensitivity Analysis

Sensitivity analysis explains a *variation* of the function, not the function value itself.

input

explanation for “car”

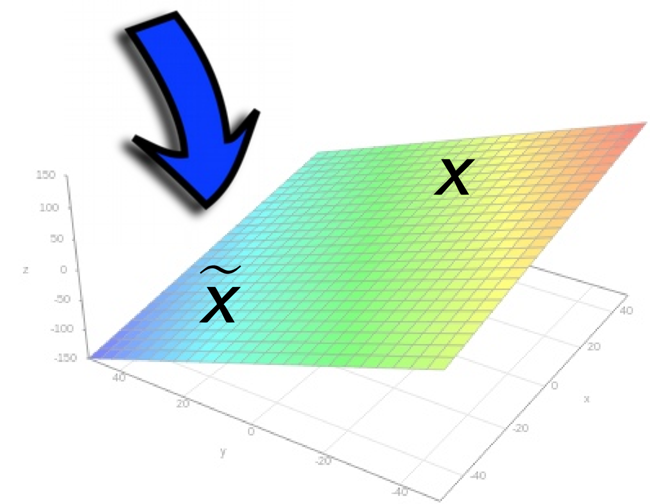


variation = make something appear less/more a car.

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

The Taylor Expansion Approach

root point



1. Take a linear model:

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} w_i x_i + b$$

2. First-order expansion at root point:

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i}\Big|_{\widetilde{\boldsymbol{x}}} \cdot (x_i - \widetilde{x}_i)$$

3. Identifying linear terms:

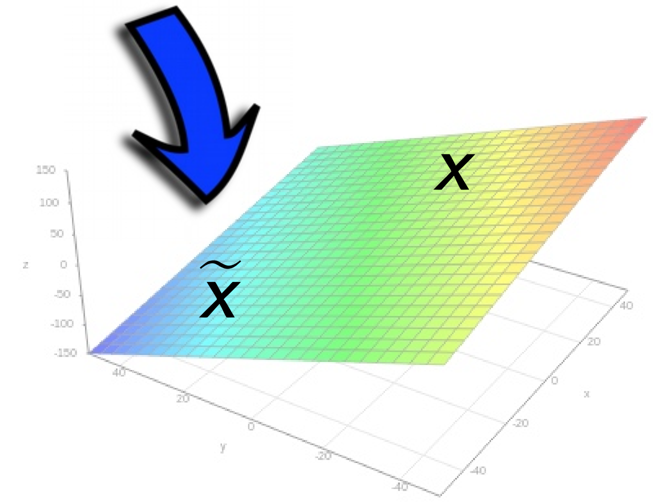
$$R_i = w_i \cdot (x_i - \widetilde{x}_i)$$

a decomposition

Observation: explanation depends on the root point.

The Taylor Expansion Approach

root point



Obtained relevance scores

$$R_i = w_i \cdot (x_i - \widetilde{x}_i)$$

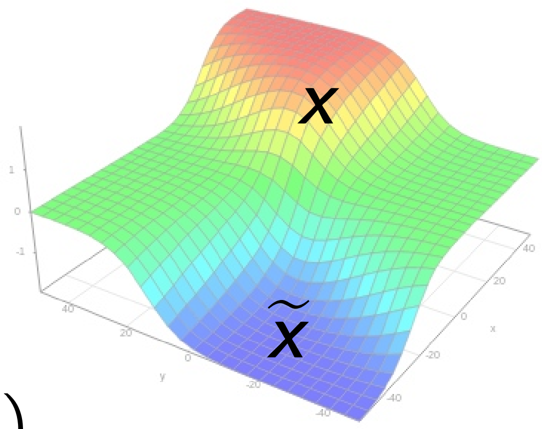
How to choose the root point ?

- Closeness to the actual data point
- Membership to the input domain (e.g. pixel space)
- Membership to the data manifold.

Non-Linear Models

Nonlinear model

$$f(\boldsymbol{x}) = \sum_j \rho\Big(\sum_{i=1}^d w_{ij} x_i + b_j \Big)$$



$$f(\boldsymbol{x}) = \sum_{i=1}^d \frac{\partial f}{\partial x_i}\Big|_{\widetilde{\boldsymbol{x}}} \cdot (x_i - \widetilde{x}_i) + o(\boldsymbol{x}\boldsymbol{x}^\top)$$

second-order terms are hard to interpret and can be very large

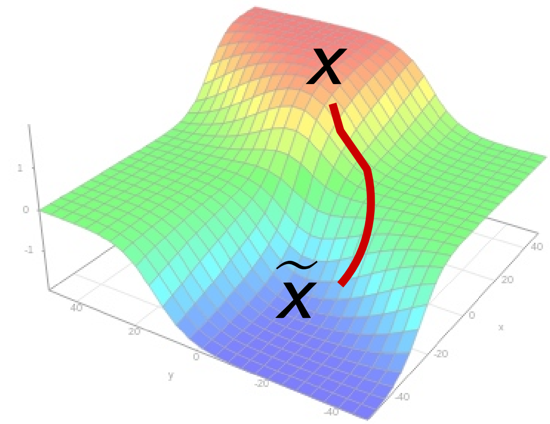
Simple Taylor decomposition is not suitable for highly non-linear models.

Overcoming NonLinearity

Integrated Gradients [Sundararajan'17]:

$$f(\boldsymbol{x}) = \int_{\boldsymbol{\xi}=\tilde{\boldsymbol{x}}}^{\boldsymbol{x}} \sum_{i=1}^{d} \frac{\partial f}{\partial x_i}\Big|_{\boldsymbol{\xi}} \cdot d\xi_i$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \underbrace{\int_{\boldsymbol{\xi}=\tilde{\boldsymbol{x}}}^{\boldsymbol{x}} \frac{\partial f}{\partial x_i}\Big|_{\boldsymbol{\xi}} \cdot d\xi_i}_{R_i}$$

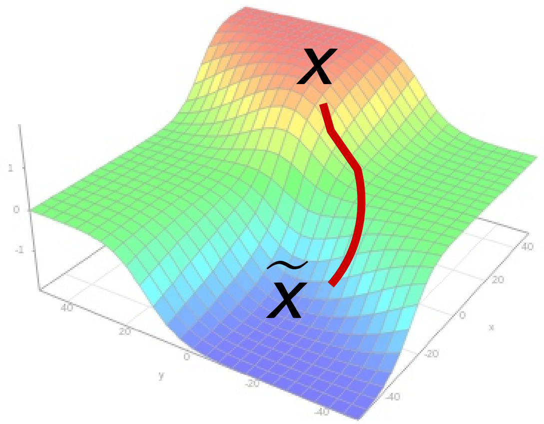


- Fully decomposable
- Require computing an integral (expensive)
- Which integration path?

[Sundararajan'17] Axiomatic Attribution for Deep Networks. ICML 2017: 3319-3328

Overcoming NonLinearity

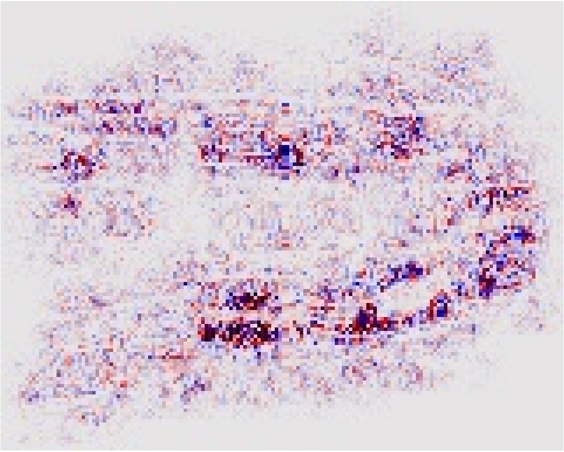
Special case when the origin is a root point and the gradient along the integration path is constant:



$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \int_{\boldsymbol{\xi}=0}^{\boldsymbol{x}} \frac{\partial f}{\partial x_i} \cdot d\xi_i$$

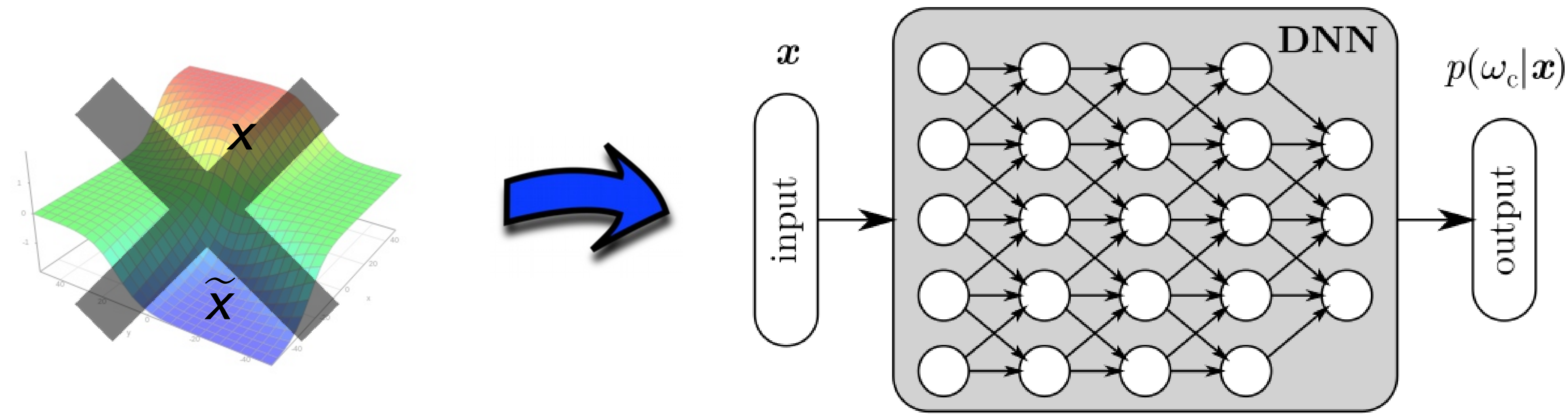
$$f(\boldsymbol{x}) = \sum_{i=1}^{d} \underbrace{\frac{\partial f}{\partial x_i} \cdot x_i}_{R_i}$$

gradient x input



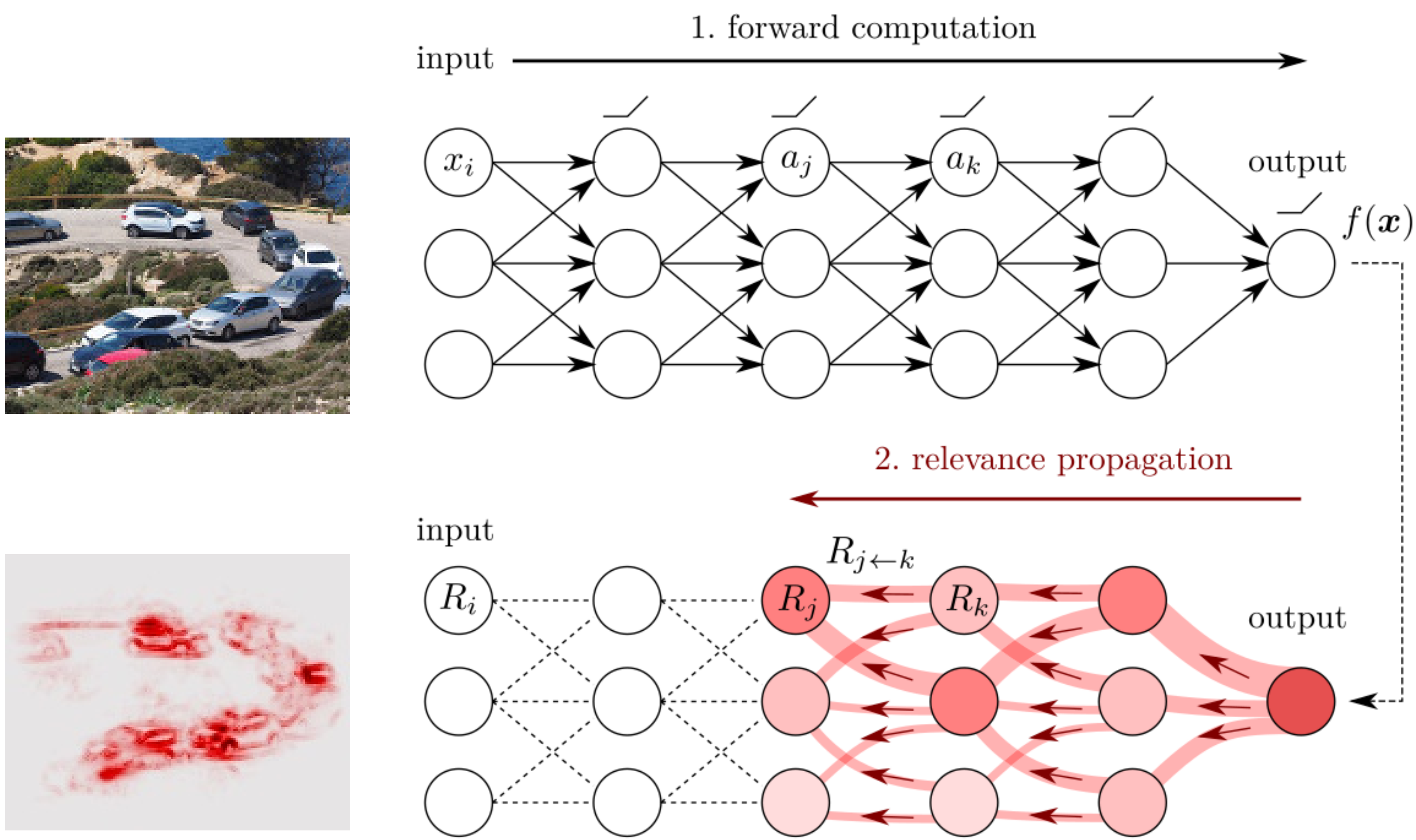
Let's consider a
different approach ...

Overcoming NonLinearity



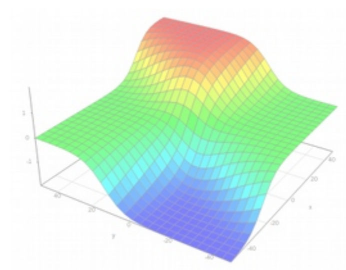
View the decision as a **graph computation** instead of a function evaluation, and propagate the decision backwards until the input is reached.

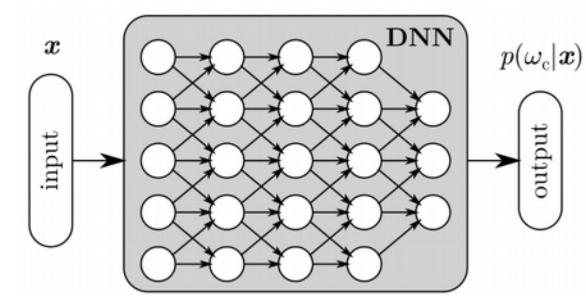
Layer-Wise Relevance Propagation (LRP) [Bach'15]



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

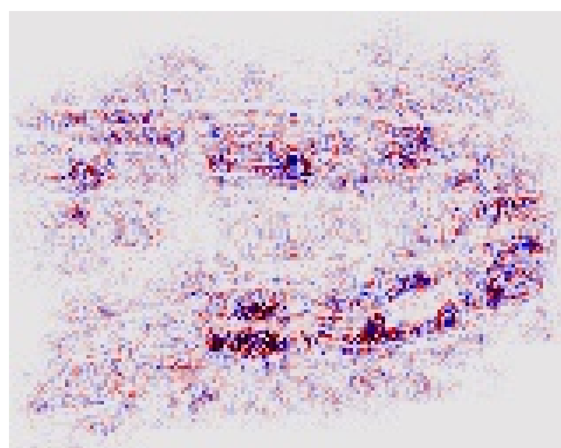
Gradient-Based vs. LRP



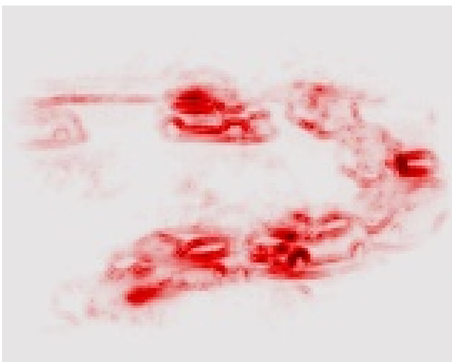


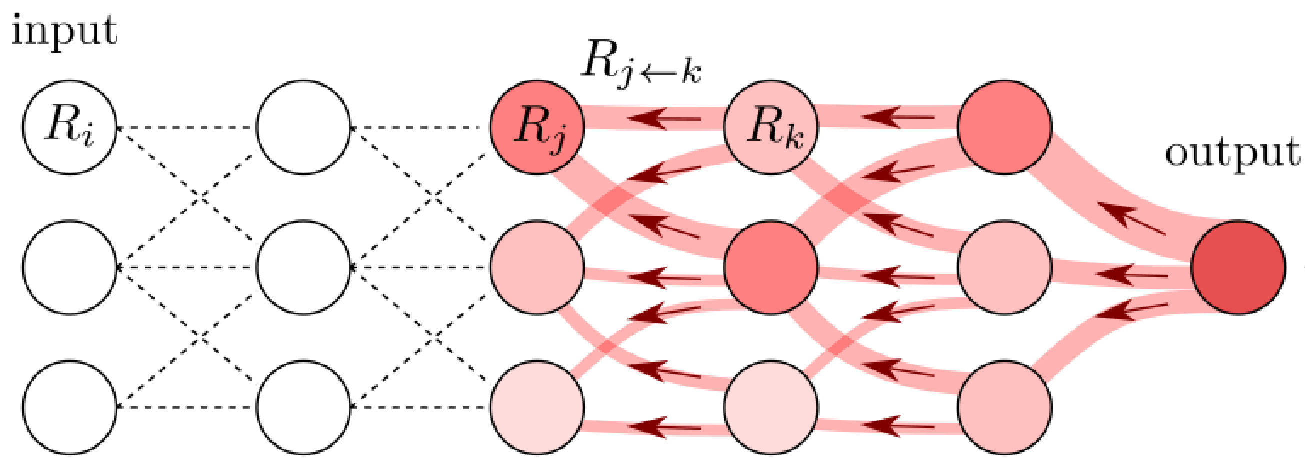
grad × input

LRP-$\alpha_1\beta_0$



Layer-Wise Relevance Propagation (LRP) [Bach'15]





Carefully engineered propagation rule:

e.g. LRP-$\alpha_1\beta_0$

neuron contribution

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

available for redistribution

pooling received messages

normalization term

LRP Propagation Rules: Two Views

View 1:

neuron contribution

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

available for redistribution

pooling received messages

normalization term

View 2:

available for redistribution

neuron activation

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

weighted sum

normalization term

Implementing Propagation Rules (1)

available for redistribution

neuron activation

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

weighted sum

normalization term

Element-wise operations	Vector operations
$z_k \to \sum_j a_j w_{jk}^{+}$	$\boldsymbol{z} \to W_{+}^{\top} \cdot \boldsymbol{a}$
$s_k \to R_k / z_k$	$\boldsymbol{s} \to \boldsymbol{R} \oslash \boldsymbol{z}$
$c_j \to \sum_k w_{jk}^{+} s_k$	$\boldsymbol{c} \to W_{+} \cdot \boldsymbol{s}$
$R_j \to a_j c_j$	$\boldsymbol{R} \to \boldsymbol{a} \odot \boldsymbol{c}$

Implementing Propagation Rules (2)

available for redistribution

neuron activation

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

weighted sum

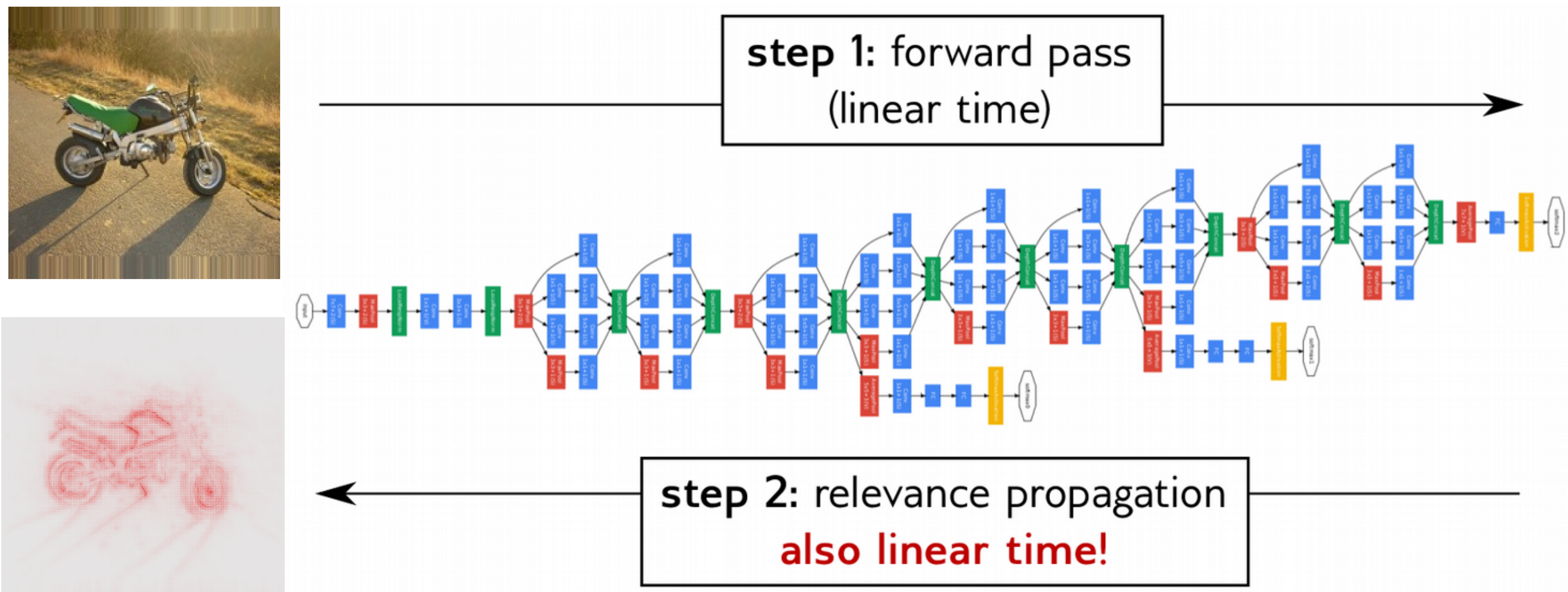
normalization term

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
```

See also http://www.heatmapping.org/tutorial

How Fast is LRP ?

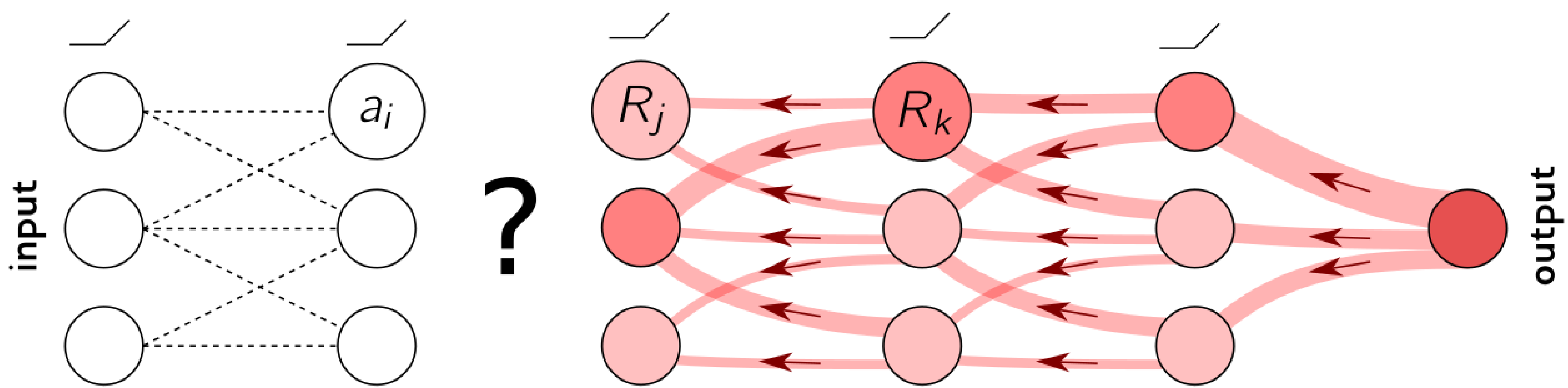


GPU-based implementation of LRP: Check out iNNvestigate [Alber'18]
https://github.com/albermax/innvestigate

Is there an underlying mathe-matical framework for LRP?

Deep Taylor Decomposition [Montavon'17]

Question: Suppose that we have propagated the relevance until a given layer. How should it be propagated one layer further?



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

Idea: By performing a Taylor expansion of the relevance.

The Structure of Relevance

Reminder:

available for redistribution

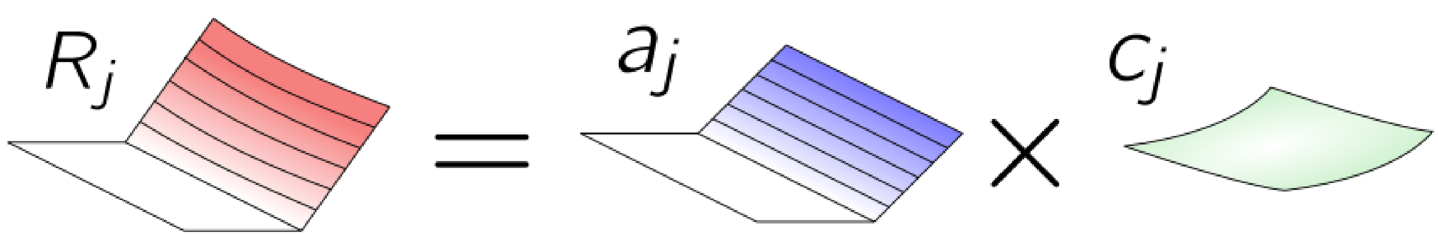
neuron activation

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

weighted sum

normalization term

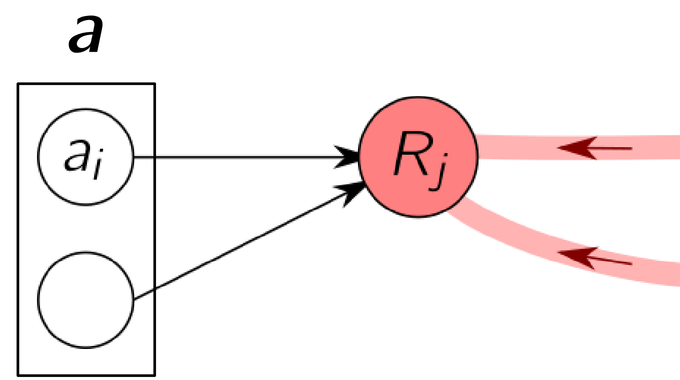
Observation: Relevance at each layer is a product of the activation and an approximately locally constant term.



Deep Taylor Decomposition

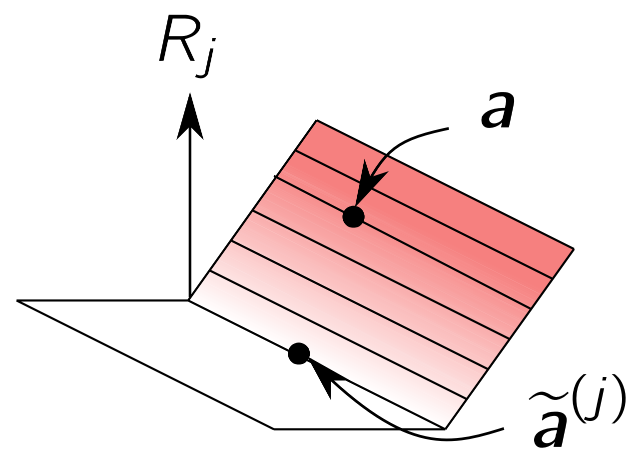
Relevance neuron:

$$R_j(\boldsymbol{a}) = \max(0, \textstyle\sum_i a_i w_{ij} + b_j) \cdot c_j$$



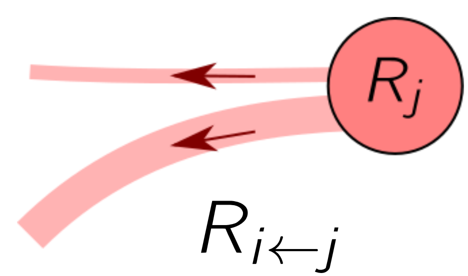
Taylor expansion:

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



Redistribution:

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



Choosing the Root Point

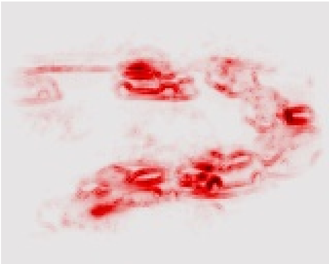
$$R_{i \leftarrow j} = \frac{(a_i - \widetilde{a}_i^{(j)}) w_{ij}}{\sum_i (a_i - \widetilde{a}_i^{(j)}) w_{ij}} R_j$$

(Deep Taylor generic)

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

(same as LRP-$\alpha_1\beta_0$)

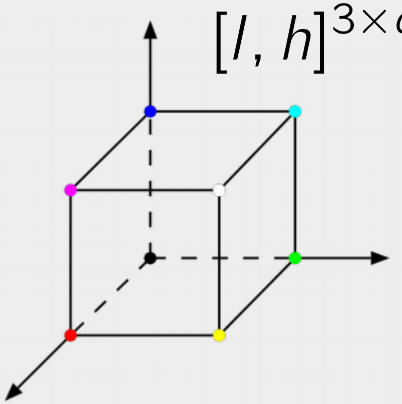
$$R_{i \leftarrow j} = \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j$$



Choosing the Root Point

$$R_{i \leftarrow j} = \frac{(x_i - \widetilde{x}_i^{(j)}) w_{ij}}{\sum_i (x_i - \widetilde{x}_i^{(j)}) w_{ij}} R_j$$ (Deep Taylor generic)

Pixels domain:



Choice of root point

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

Resulting propagation rule

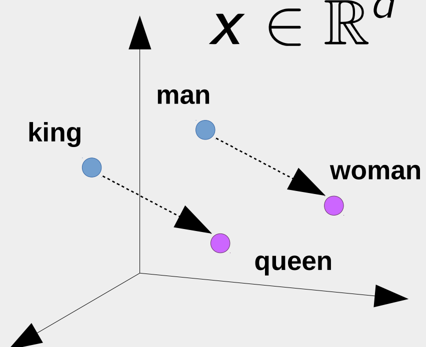
$$R_{i \leftarrow j} = \frac{x_{ij} w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_{ij} w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j$$

Choosing the Root Point

$$R_{i\leftarrow j} = \frac{(x_i - \widetilde{x}_i^{(j)})w_{ij}}{\sum_i (x_i - \widetilde{x}_i^{(j)})w_{ij}} R_j$$ (Deep Taylor generic)

Word embedding:

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



adapted from
Tensorflow tutorial

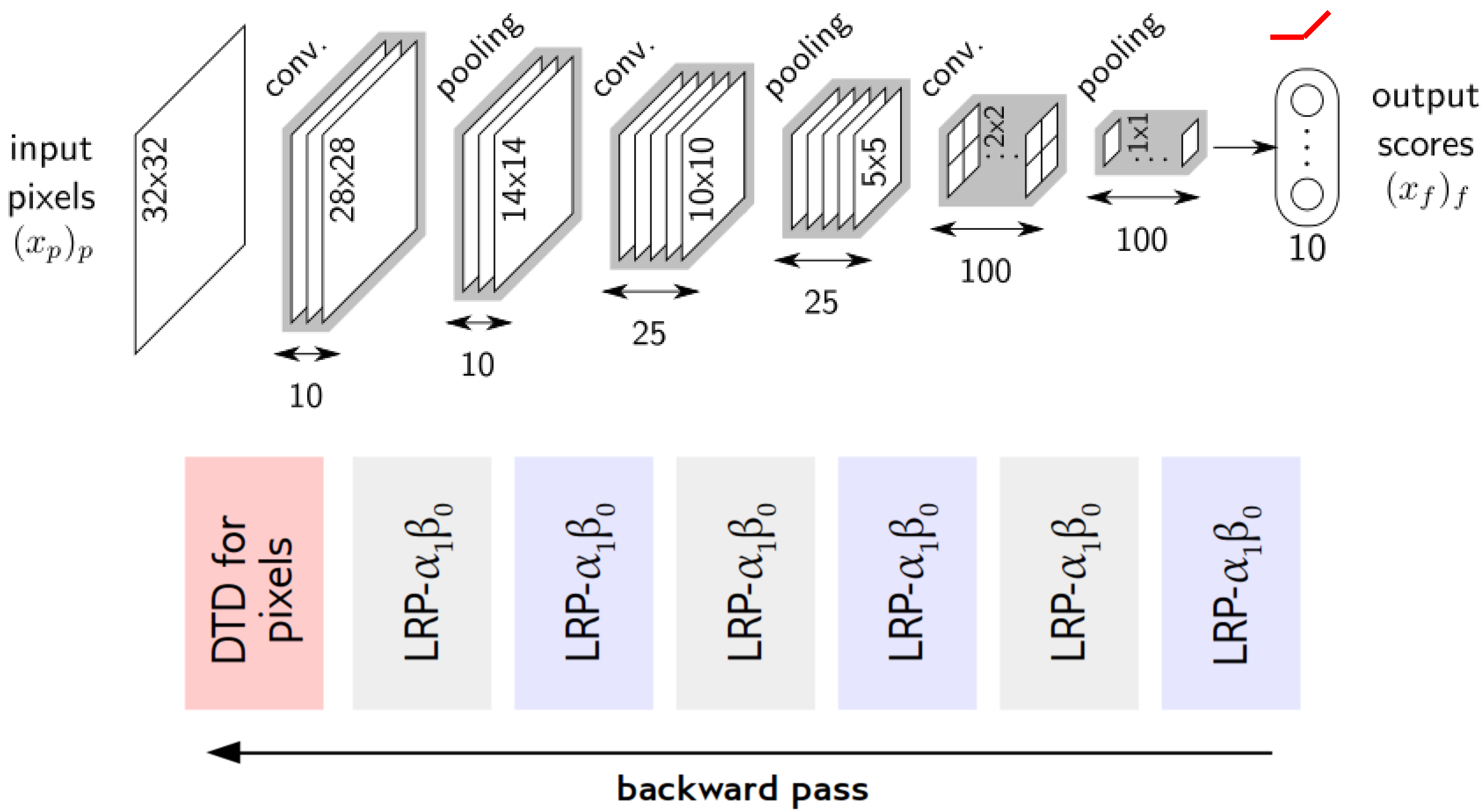
Choice of root point

$$\widetilde{\boldsymbol{x}}^{(j)} = \boldsymbol{x} - t \cdot \boldsymbol{w}_j$$

Resulting propagation rule

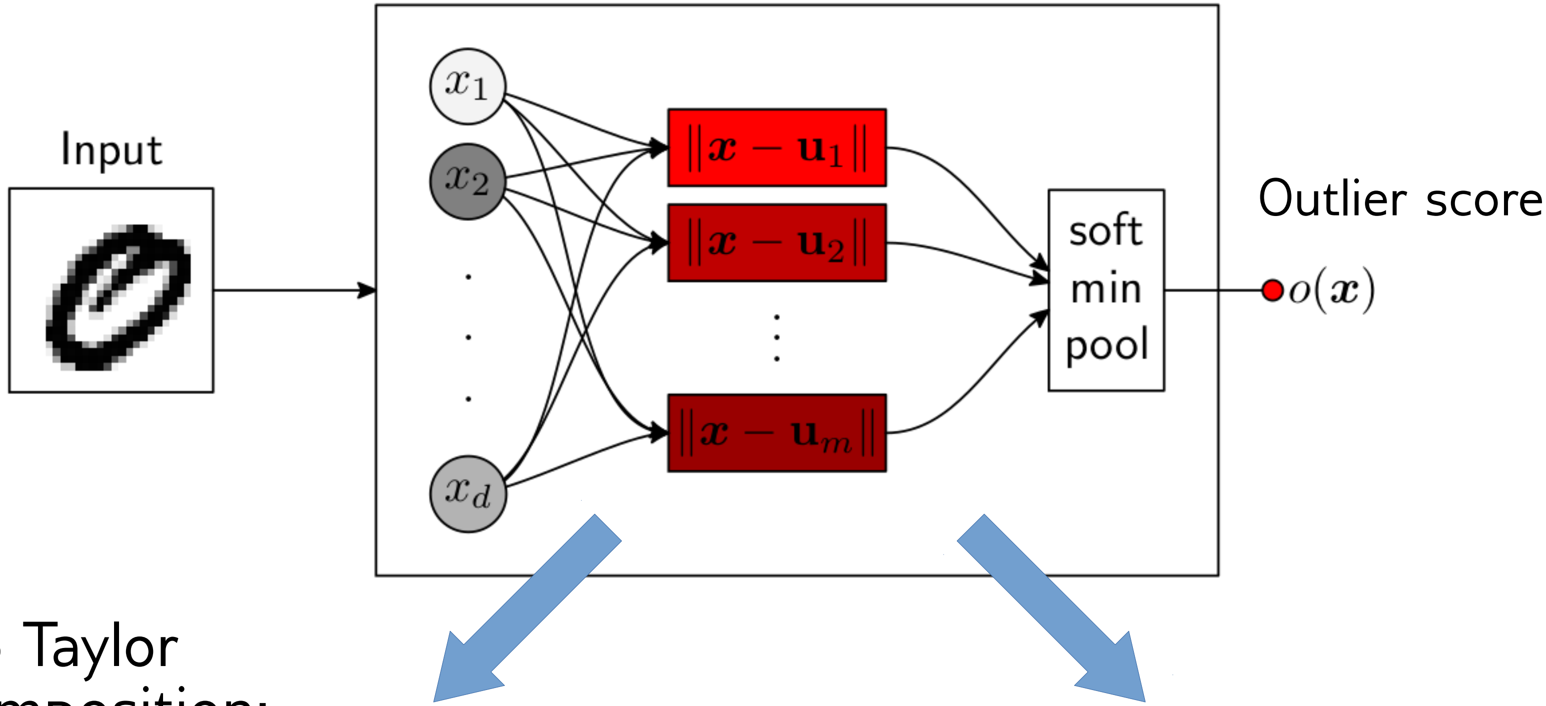
$$R_{i\leftarrow j} = \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j$$

DTD View on Explaining a ConvNet [Montavon'17]



DTD View on Explaining an OCSVM [Kauffmann'18]

One-class SVM rewritten as a min-pooling over distances:

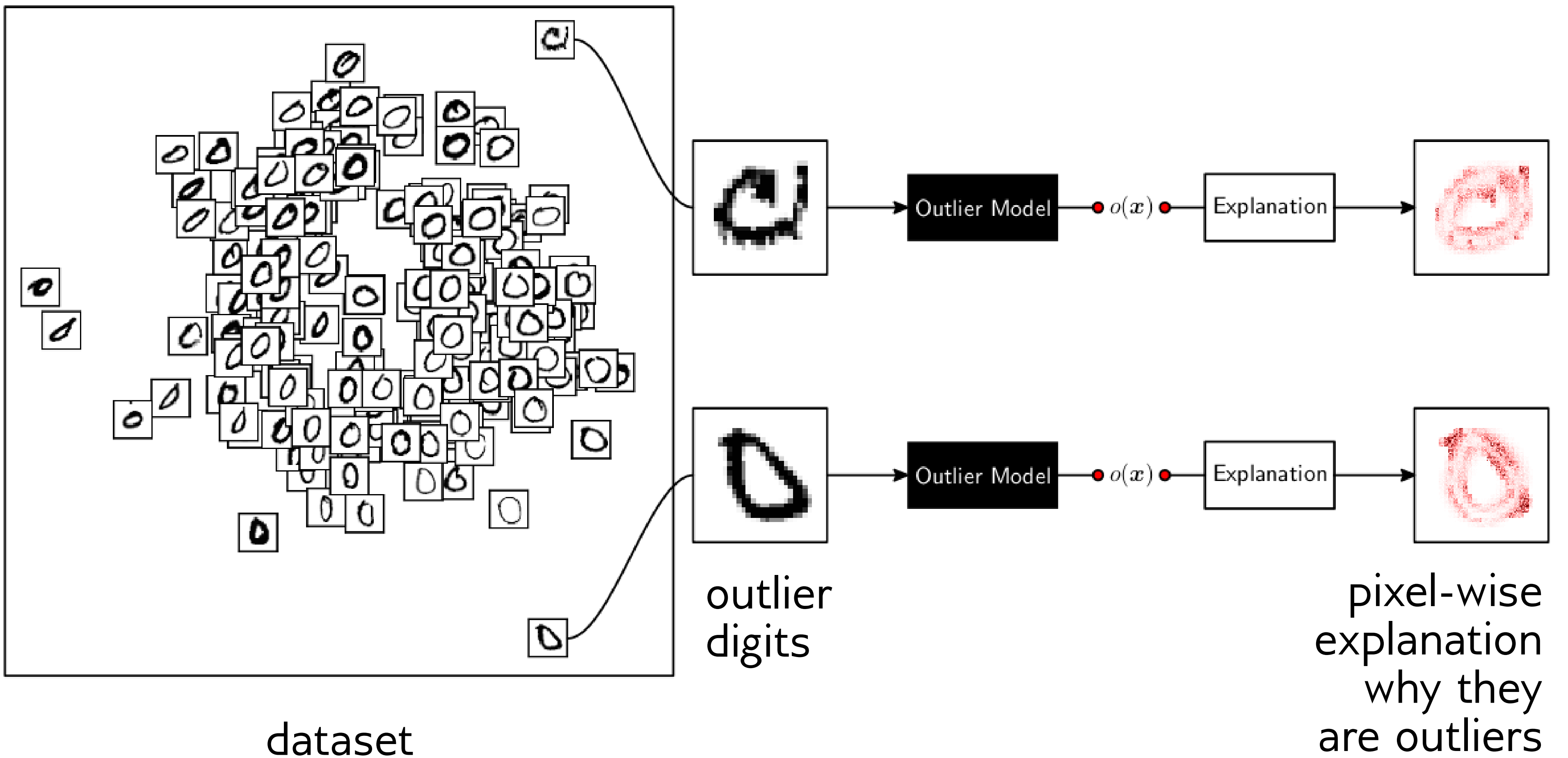


Deep Taylor decomposition:

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

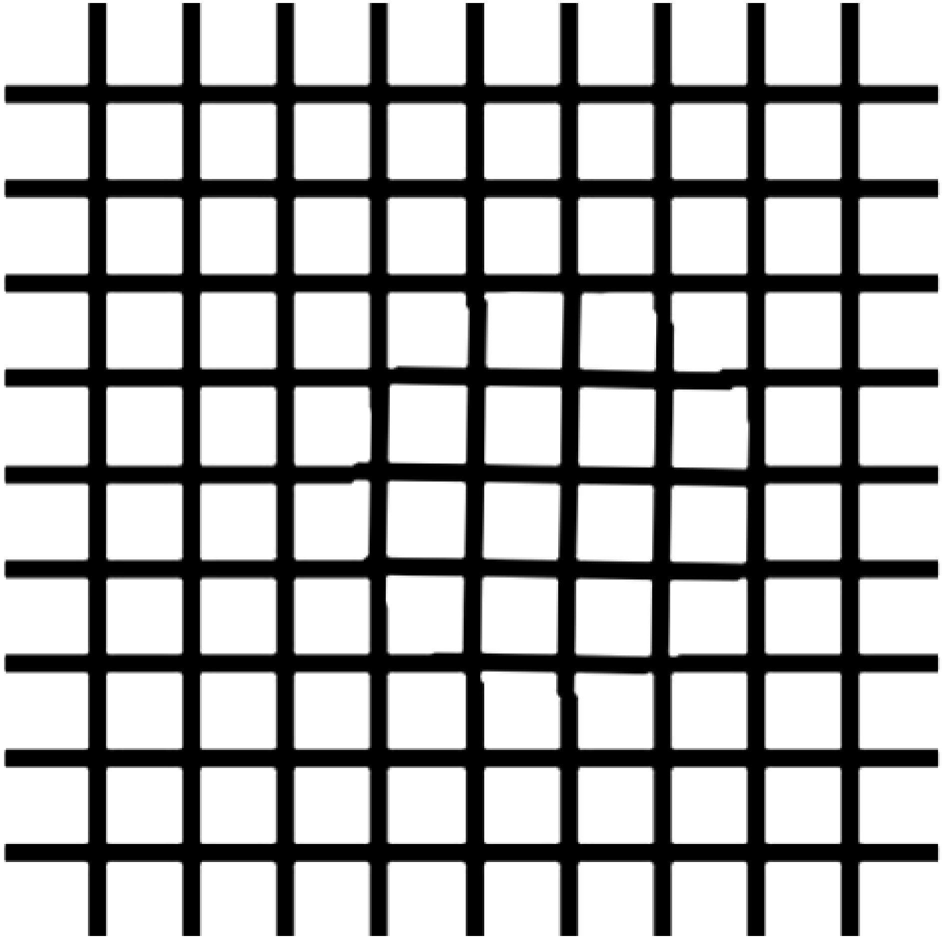
$$R_j = (a_j + \varepsilon_j) \cdot \frac{\exp(-a_j)}{\sum_j \exp(-a_j)}$$

DTD-OCSVM on MNIST

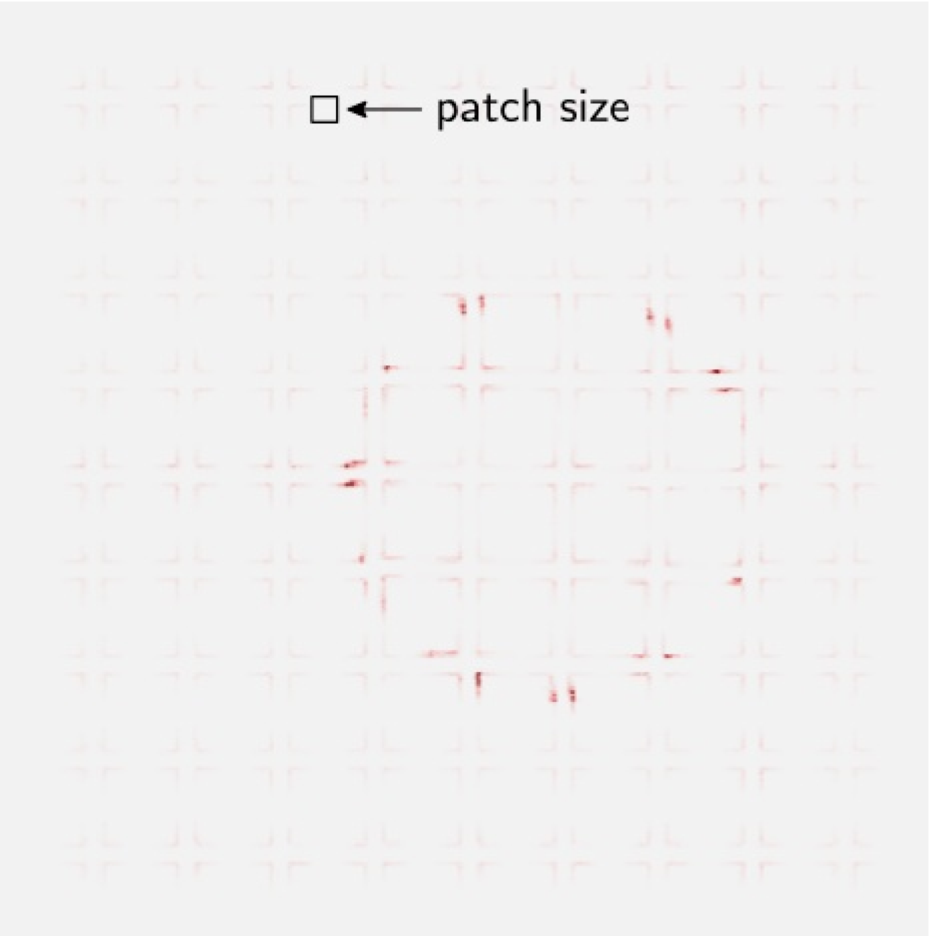


DTD-OCSVM on Images

input



explanation for outlierness



Conclusion for Part 2

1. Explaining deep neural networks is non-trivial. Simple gradient-based methods either do not ask the right question, or are difficult to scale to deep models.

2. Propagation-based approaches (e.g. LRP) seem to work better on complex DNN models. (This will be validated in Part 3).

3. Deep Taylor Decomposition provides a theoretical framework for understanding and deriving LRP-type explanation procedures.