Aspects of explainable AI

Alexander Binder

University of Oslo (UiO)

April 19, 2022



UiO : Department of Informatics
University of Oslo

In this lecture we can cover only a few aspects (a lecture into coding LRP for pytorch takes me alone 1 hour!).

- model interpretation (two approaches)
  - t-SNE embeddings (pytorch + scikit learn): show similarities for features from a dataset
  - compute which inputs activate a feature most
- decision interpretation (one unlabeled test sample as compared to a dataset)
  - Shapley values
  - Linearization-based ideas: Gradient, sensitivity, gradient times input, Grad-CAM, guided backprop, LRP and Deep Taylor
  - applications to finding biases in your data, to identify systematic failcases

Learning goals I

- an overview of different questions in explainability
- some selected aspects:
  - low-dimensional embeddings of a set of samples: t-SNE
  - samples maximizing activations in a feature layer
  - nearest examples with respect to a metric defined by a feature layer
  - applications to finding biases in your data, to identify systematic failcases

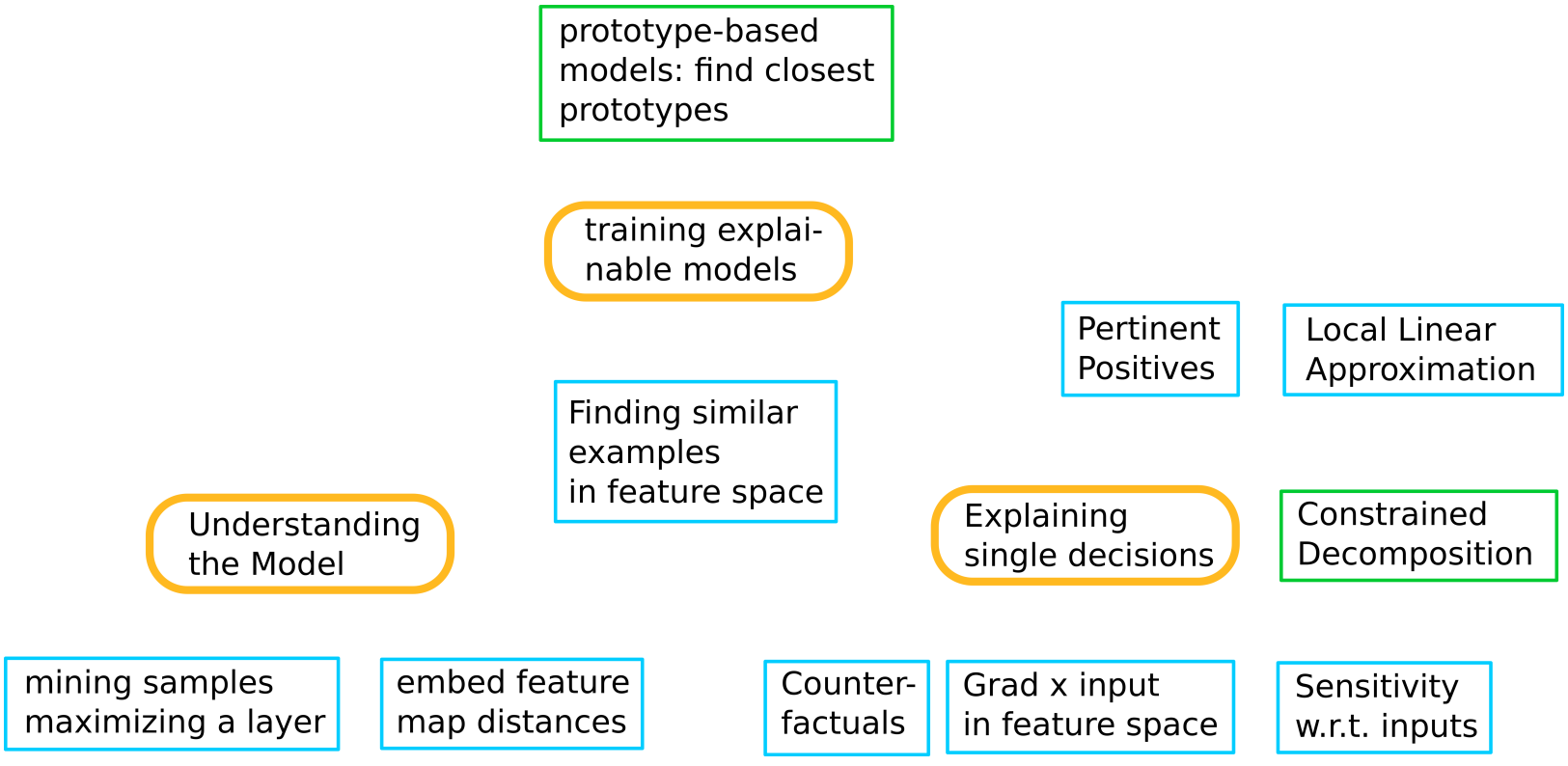
Learning goals II

- an overview of different questions in explainability
- some selected aspects:
  - explanations of a single decision
    - norm of gradients, gradient times input, integrated gradient, Grad-CAM, Occlusion, Guidedbackprop, LRP

You do not need to memorize complex LRP or t-SNE formulas for the exam, only simple formulas like gradient times input

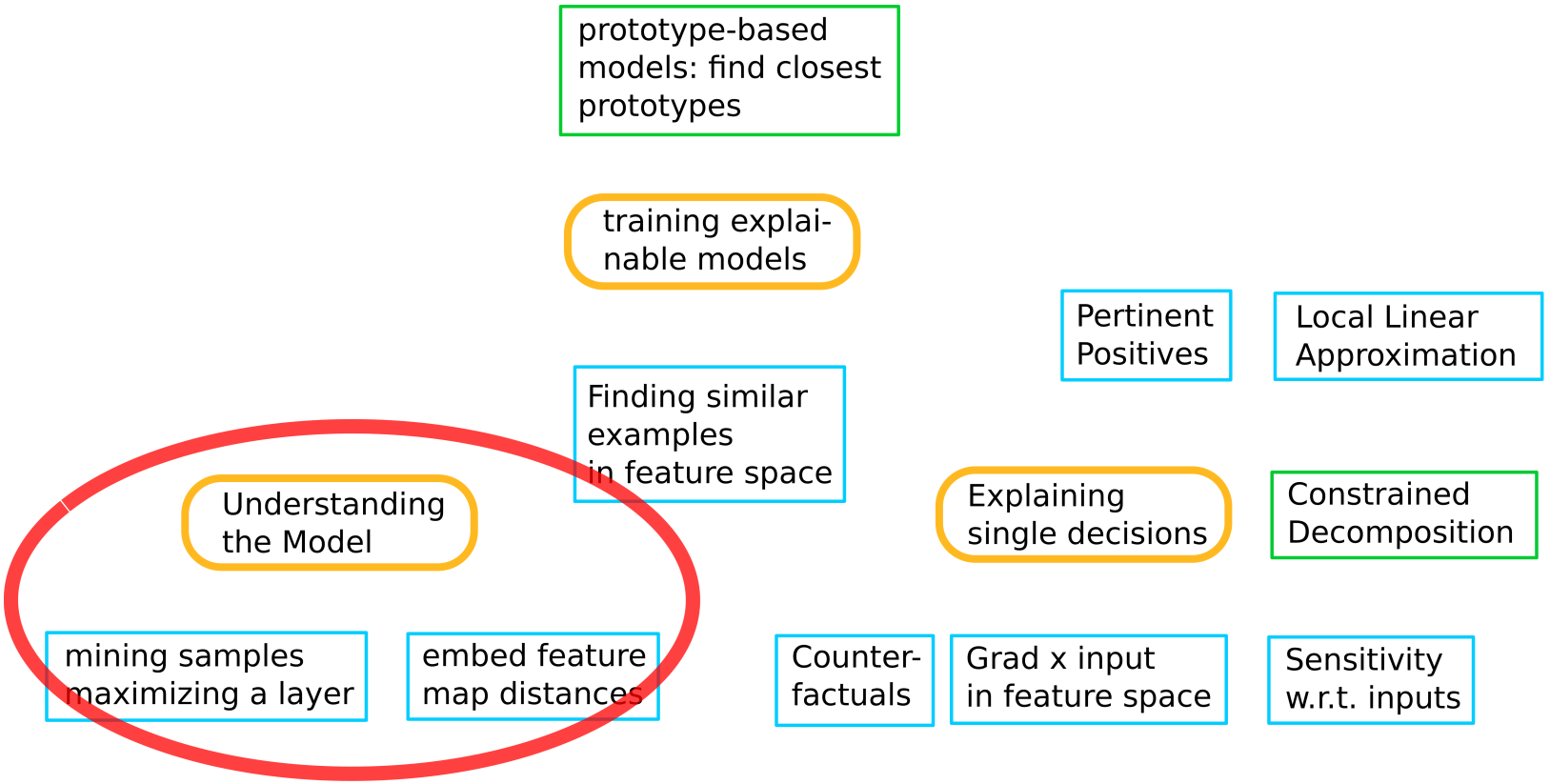
1. The definition of an explanation depends on the question

1. As many definitions of explanation as there are different questions



Authors opinion: no method rules them all

1. As many definitions of explanation as there are different questions



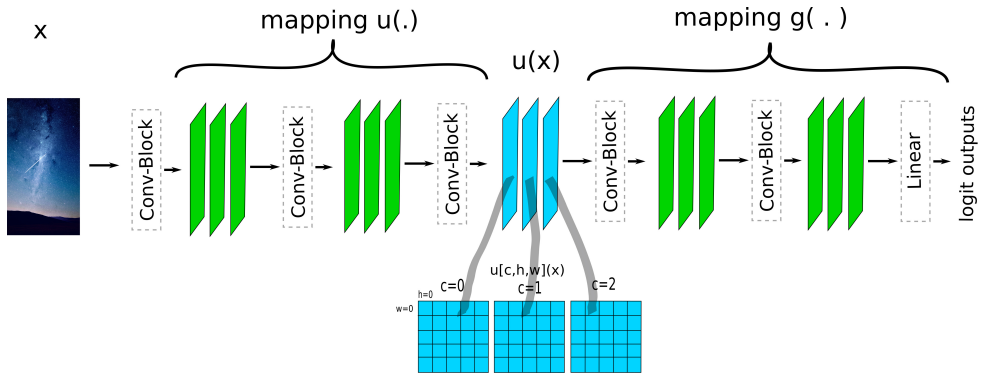
Authors opinion: no method rules them all

Outline

1. Model interpretation
2. Decision interpretation

Define low-dimensional embeddings which approximate high-dimensional distances of a feature map

Popular in deep learning: t-SNE
van der Maaten et al.:
https://lvdmaaten.github.io/tsne/examples/caltech101_tsne.jpg



- take any feature map of a neural network $u[0, c, h, w](x)$ which depends on an input sample x.

  Compute distances between different samples x_i, x_j:

$$d_u(x_i, x_j) = \|u[0, c, h, w](x_i) - u[0, c, h, w](x_j)\| = \|u_i - u_j\|$$

- Idea: visualize distances $d_u(x_i, x_j)$ over a set of samples $x_i, x_j, x_k, x_l, x_m, \ldots$ by low-dimensional approximation

Suppose you have feature maps of 200 images, how to visualize the similarities between these feature maps ?

Technical problem: how to plot a number of d-dimensional features in 2-dims such that the distances are meaningfully preserved?

One way is to visualize the similarities between samples by projecting each $u_i \in \mathbb{R}^d$ into 2 dimensions – according to their similarities and look at them. One way is t-SNE (L. van der Maaten): http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

https://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html#sphx-glr-auto-examples-manifold-plot-lle-digits-py
Working principle: Given high dimensional data points $D_n = (u_1, \ldots, u_n)$, goal is to map each $u_i \in D_n$ onto a 2-dimensional data point y_i such that y_i, y_j which have similar distances to each other as the samples u_i, u_j from the set D_n.

- given two samples i, j with features u_i, u_j
- step 1: compute the probability that i would vote for j as being his neighbor based on a gaussian model which is centered on x_i as mean

$$p_{j|i} \propto \exp(-\|u_j - u_i\|^2/2\sigma^2) \text{ and just normalize this!}$$

$$p_{j|i} = \frac{\exp(-\|u_j - u_i\|^2/2\sigma^2)}{\sum_{k:k\neq i} \exp(-\|u_i - u_k\|^2/2\sigma^2)} \Rightarrow \sum_j p_{j|i} = 1$$

- symmetrize:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \; p_{ii} = 0$$

Reason? Ensures that $\sum_i p_{ij} > \frac{1}{2n}$, so each point, even an outlier has some large interactions to his neighbors. Otherwise points i which are very far outliers may contribute little to the embedding because $p_{i|j} \approx 0$.

What did we obtain?

- p_{ij} is a model of interaction strength between two samples with features u_i, u_j

next step:

- define a model of interaction strength $q_{ij} = q(y_i, y_j)$ between low-dimensional representatives y_i and y_j.
- Idea: tune parameters of $q(\cdot)$ until $q_{ij} \approx p_{ij}$
- result: get for u_i a synthetic sample $y_i \in \mathbb{R}^2$

- learn a similar, but heavy-tailed distribution model of y_i voting for y_j as neigbor:

$$q_{ij} \propto (1 + \|y_i - y_j\|^2)^{-1} \text{ and normalize again}$$
$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k,l:k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

- how to optimize for q_{ij}? Minimize Kullback-Leibler-Divergence:

$$\{q_{ij} \mid (i,j)\} = \text{argmin}_{\{q_{ij}\}} KL(P||Q) = \text{argmin}_{\{q_{ij}\}} \sum_{ij} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

- minimize for y_i by computing the gradient of $KL(P||Q)$ with respect to y_i (P is fixed, Q depends on y_i)

Why for the y_i use a heavy tailed probability?

https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf:
"This allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance in the map and, as a result,it eliminates the unwanted attractive forces between map points that represent moderately dissimilar datapoints."

Idea is the following: in high dimensional spaces many points can have intermediate distance to each other, resulting in an intermediate interaction probability p_{ij}.

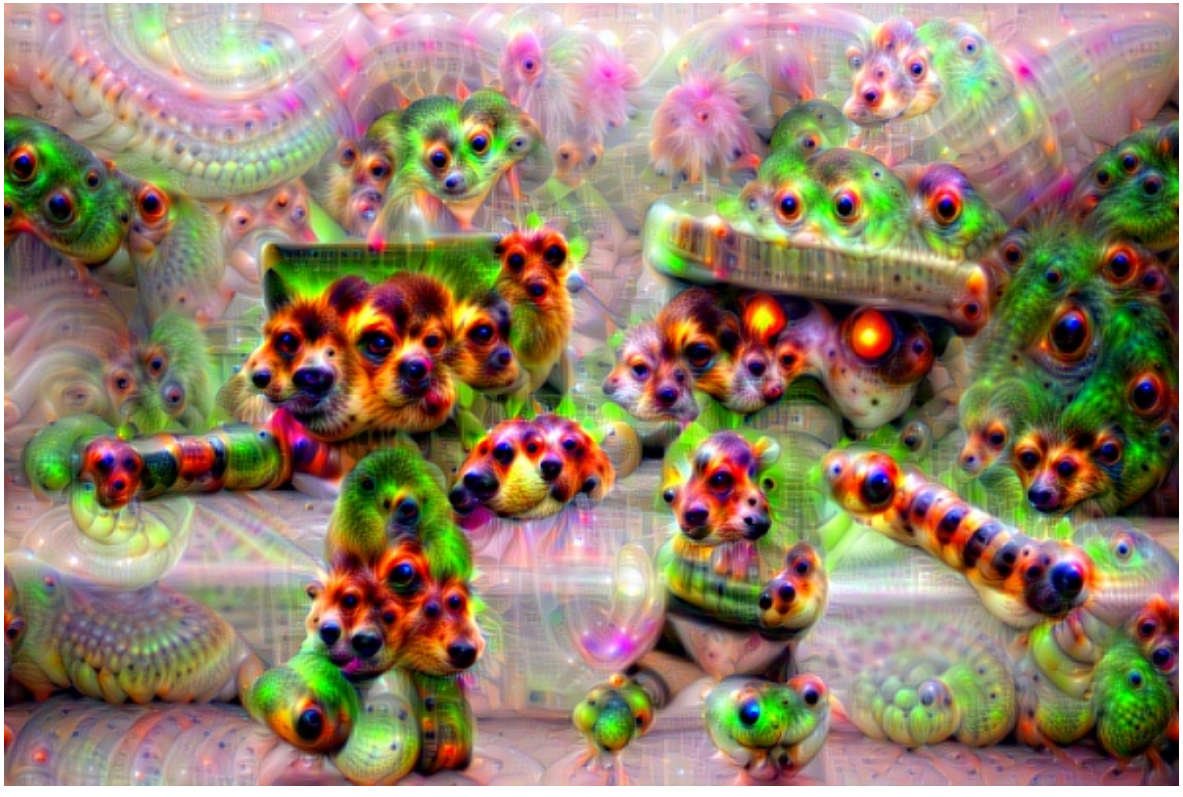
A heavy tailed probability assigns a relatively high probability to points far away. Therefore those points u_i, u_j with intermediate distance in the original space can be assigned to points y_i, y_j in the 2-d model which are far away (and still result in an intermediate-valued q_{ij} which fits well to the intermediate-valued p_{ij}.)

By this t-sne can focus on putting only those points i and j close in the embedding for which the original points x_i and x_j are really close (and have a high p_{ij}).

Properties and limitations of t-SNE:
https://distill.pub/2016/misread-tsne/ – what one gets out from t-sne, depends a lot on the perplexity parameter choice, and it may be very different from the original distances. Its a visualization, not some kind of truth. Results need to be validated.

- IDEA: visualize distances by low-dimensional approximation
- parameter sensitivity https://distill.pub/2016/misread-tsne/
- good for exploration but requires a follow up confirmation of findings
- CHAL: how to choose a low-dimensional approximation? Is there the one?
- alternatives: PCA projections (K. Pearson), Isomap (Tenenbaum et al. graph defined by k-nearest neighbors and euclidean distances along edges), many others

DeepDream as an example of Activation Maximization



Credit: https://github.com/gordicaleksa/pytorch-deepdream

In what ways can one enhance it with more than esthetic value?

simple idea:

- choose a feature map channel $f[0, c, h, w]$ and a statistic of a feature map, for example

$$s_c(x) = \sum_{h,w} \max(0, f[0, c, h, w])$$

- the feature map statistic depends on the input image x
- do gradient-based optimization of x to maximize s_c - similar as for adversarial attacks

Finding similar examples for a given sample

- select a layer/feature map $h(\cdot)$
- given a sample x and its feature map $h(x)$, find the closest examples in a test set, given a norm $\|\cdot\|$ or metric $d(h(x), h(x_i))$

$$\text{sort } x_i \text{ according to } \|h(x_i) - h(x)\|$$

- PRO: uses learned similarities around x. $h(\cdot) = h_w(\cdot)$. w are network parameters
- limitation: if $\|h(x_i) - h(x)\|$ is small in a layer, it does not guarantee that the final prediction will be similar for x_i and x or that the internal reasoning will be similar for x_i and x. Could diverge in higher layers! Extreme case: $h(\cdot) =$ input layer and adversarials.
- difference to t-SNE: this tries to explain similarities to a single sample x as defined by a chosen feature map, not visualize a set of samples or what activates a feature map most

Understanding the model: Mining samples which maximize a channel activation

The top-3 images which maximally activate a particular channel of layer `layer4.2.conv2` of a ResNet-50 after fine-tuning on Pascal VOC. Selection within the Pascal VOC validation set. The picture also shows an explanation which region in the image is contributing to the activation of the channel (LRP explanation from the maximally activated spatial element).



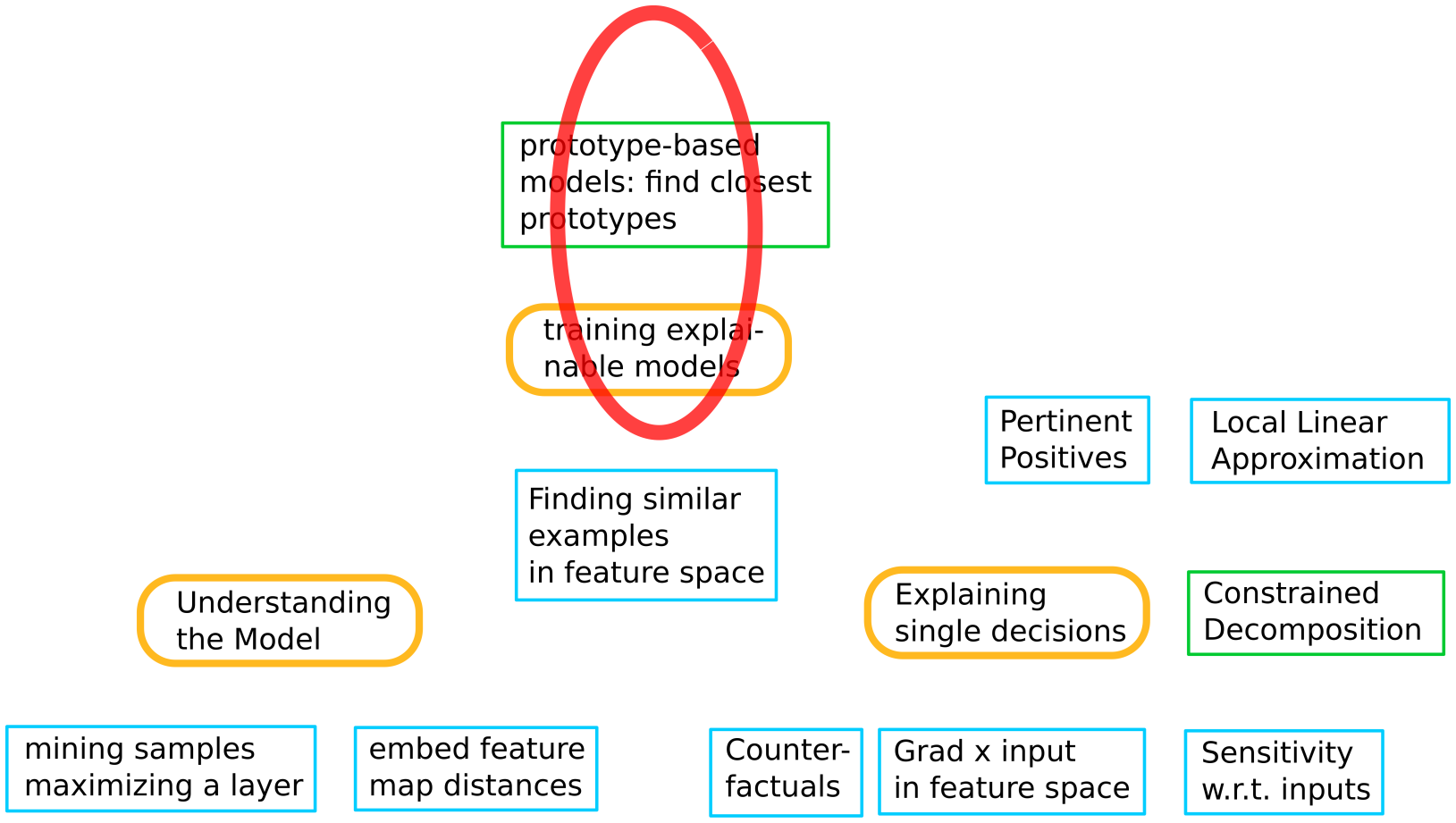
Channel has learnt to detect bus views.

1 Model interpretation

2 Decision interpretation

Delta to before ?
have a single sample x and a prediction $f(x)$. Define measures how x is important to arrive at $f(x)$.

1. The definition of an explanation depends on the question



Authors opinion: no method rules them all

Understanding a *single* prediction: Prototype-based learning

Find most similar samples that were used to arrive at a prediction for a sample x.

- k-nearest neighbors
- What classifier can be expressed like this ?

$$f(x) = \sum_i \alpha_i y_i k(x_i, x)$$

Find most similar samples that were used to arrive at a prediction.

This Looks Like *That*: Deep Learning for Interpretable Image Recognition

Chaofan Chen*
Duke University
cfchen@cs.duke.edu

Oscar Li*
Duke University
oscarli@alumni.duke.edu

Chaofan Tao
Duke University
chaofan.tao@duke.edu

Alina Jade Barnett
Duke University
abarnett@cs.duke.edu

Jonathan Su
MIT Lincoln Laboratory†
su@ll.mit.edu

Cynthia Rudin
Duke University
cynthia@cs.duke.edu

Credit: Chen et al. https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf

Find most similar samples that were used to arrive at a prediction.

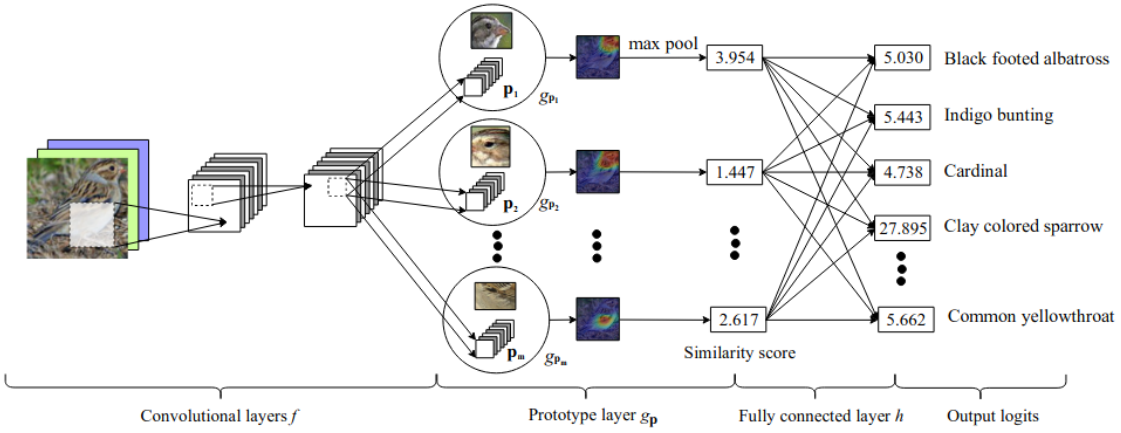
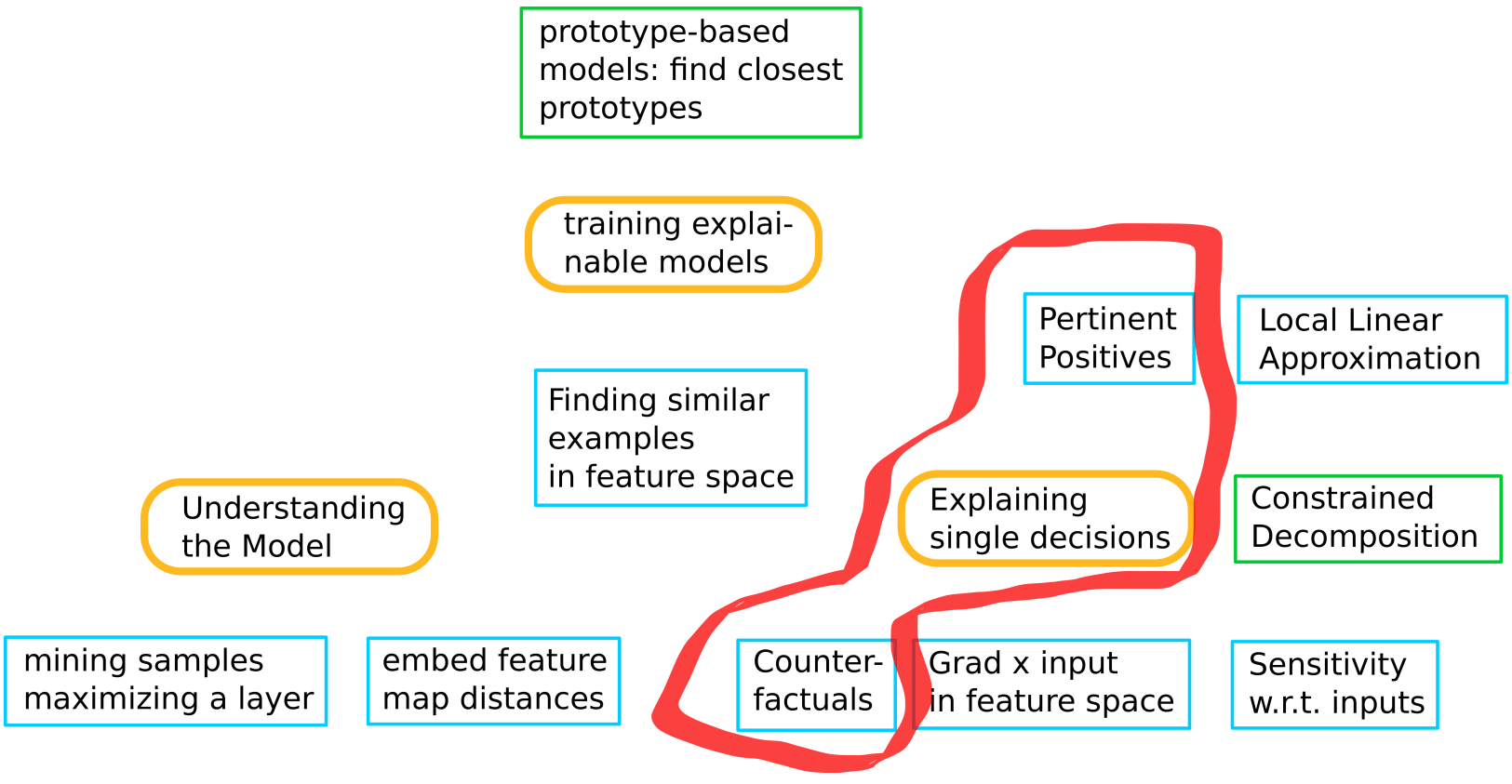


Figure 2: ProtoPNet architecture.

Credit: Chen et al. https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf

- CHAL: how to choose prototypes without combinatorial issues?
- Prototypes learned or from training data?
- semantic gap: input space versus feature space similarities https://arxiv.org/abs/2105.02968 – sensitivity to JPEG artefacts
- scalability (intra-class and inter-class)?

1. The definition of an explanation depends on the question



Authors opinion: no method rules them all

Understanding a *single* prediction: Pertinent positives and Negatives

Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives

Amit Dhurandhar*
IBM Research
Yorktown Heights, NY 10598
adhuran@us.ibm.com

Pin-Yu Chen*
IBM Research
Yorktown Heights, NY 10598
pin-yu.chen@ibm.com

Ronny Luss
IBM Research
Yorktown Heights, NY 10598
rluss@us.ibm.com

Chun-Chen Tu
University of Michigan
Ann Arbor, MI 48109
timtu@umich.edu

Paishun Ting
University of Michigan
Ann Arbor, MI 48109
paishun@umich.edu

Karthikeyan Shanmugam
IBM Research
Yorktown Heights, NY 10598
karthikeyan.shanmugam2@ibm.com

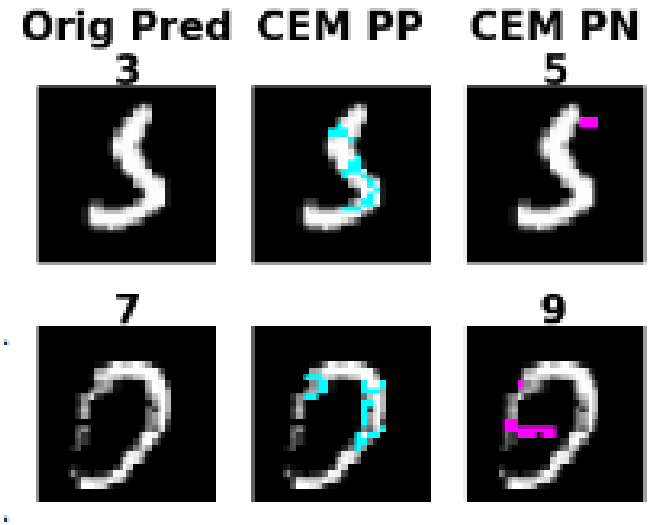
Payel Das
IBM Research
Yorktown Heights, NY 10598
daspa@us.ibm.com

Credit: https://proceedings.neurips.cc/paper/2018/file/c5ff2543b53f4cc0ad3819a36752467b-Paper.pdf

Understanding a *single* prediction: Pertinent positives and Negatives

Pertinent positive: what to retain from a sample?

Pertinent negative: what to change so that prediction switches?



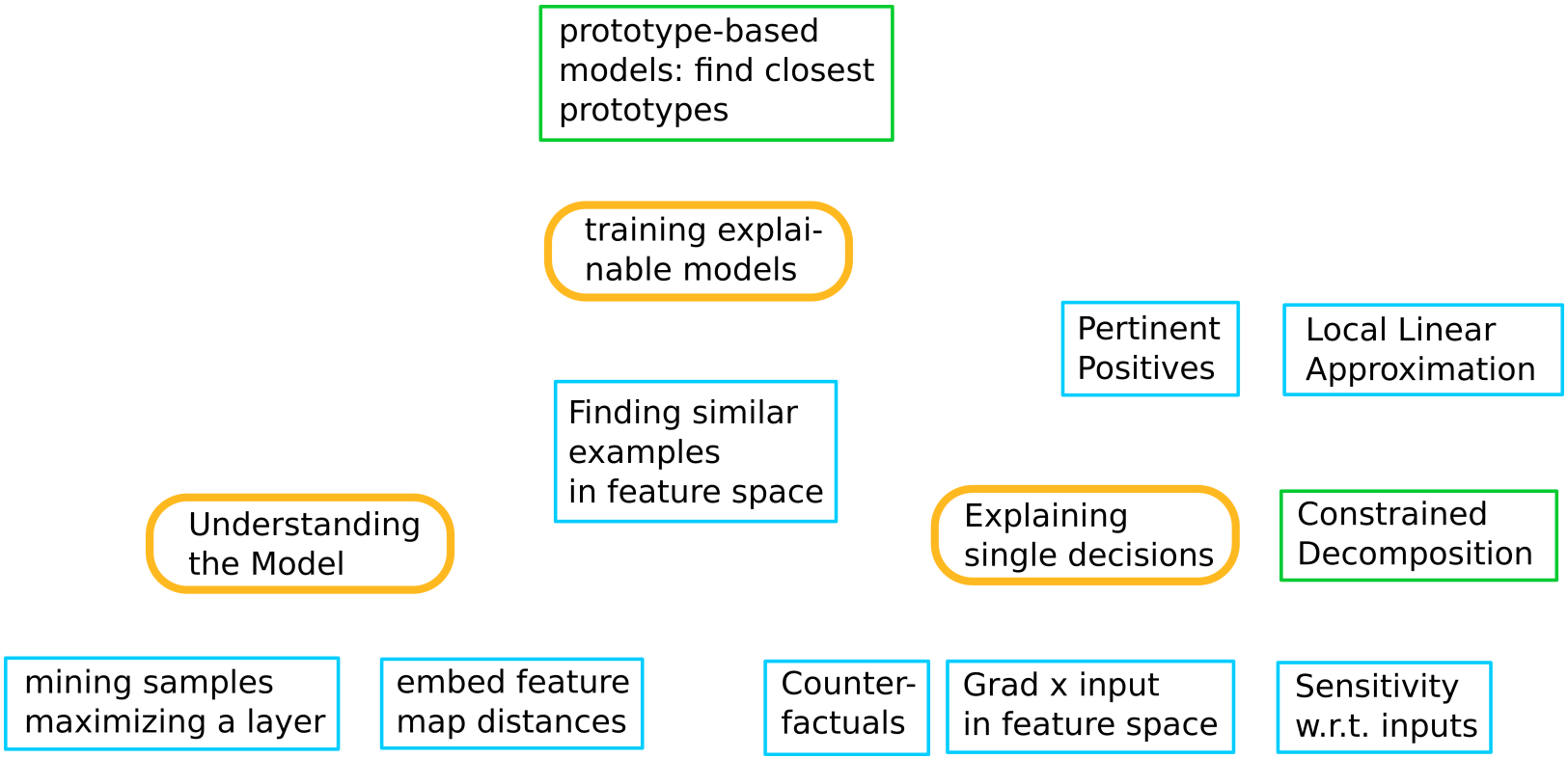
Pertinent positive: cyan, pertinent negative: pink

Credit: https://proceedings.neurips.cc/paper/2018/file/c5ff2543b53f4cc0ad3819a36752467b-Paper.pdf

- CHAL: how to delete/replace information? Result is plausible/outlier?
- many different PP/PN – how to integrate them?

Is this a complete picture?

Many ways to define an explanation of a prediction or a model.





Am I one of them?

Credit: Hanabusa Itcho

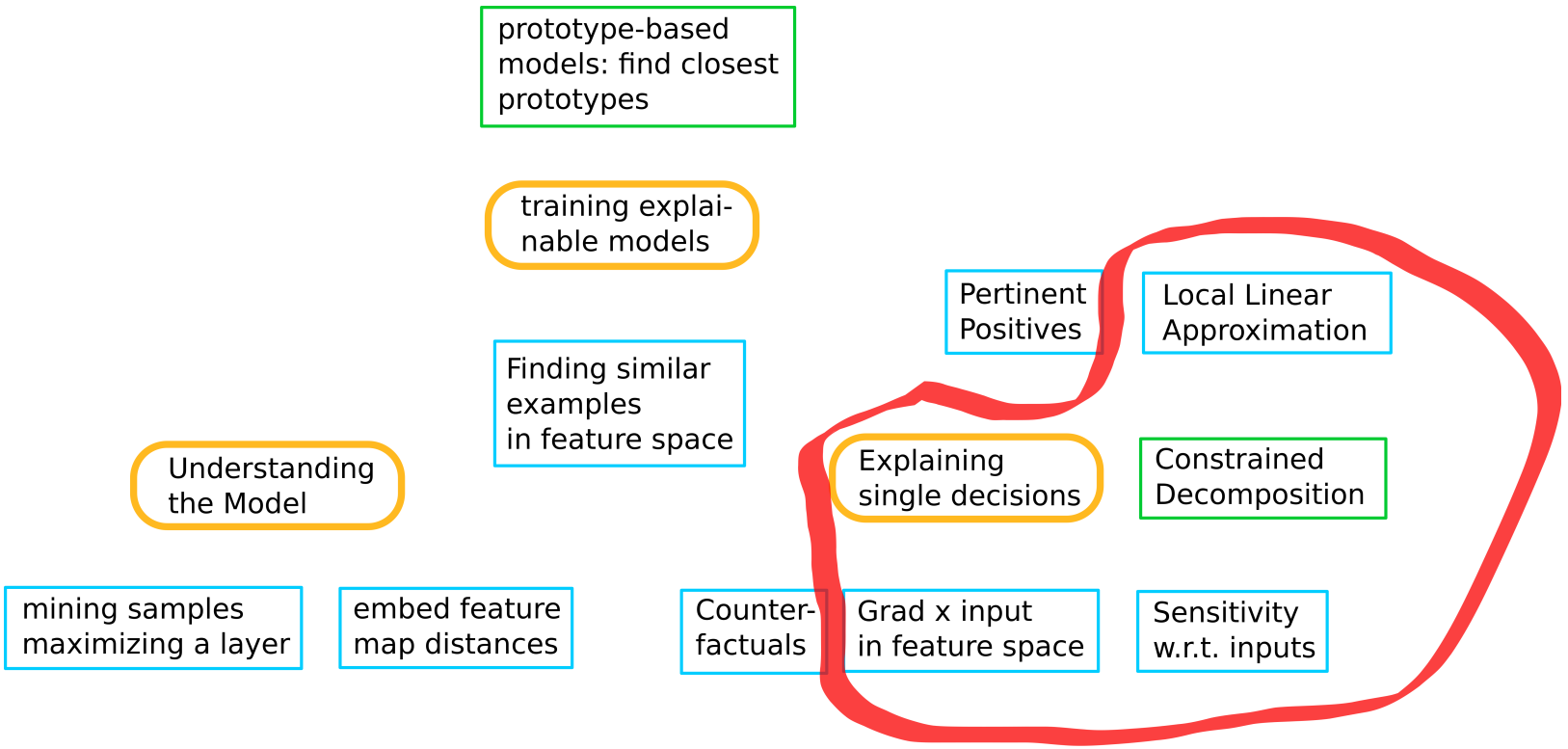
One can take it further in a human-machine interaction context in collaboration with HCI experts:

> A dialogue between the human and the system.
>
> Possibly open-ended question answering?
>
> A system which can rephrase the question, query on uncertain parts with counter questions.
>
> System brings evidence when providing an answer – Explanations as a reasoning problem with self-awareness of ones internals.

- A path from explanation to reasoning about oneself.
- One possible goal of building general intelligent systems.

A more narrow scope: explaining predictions on a single sample by decomposition

Many ways to define an explanation of a prediction or a model.

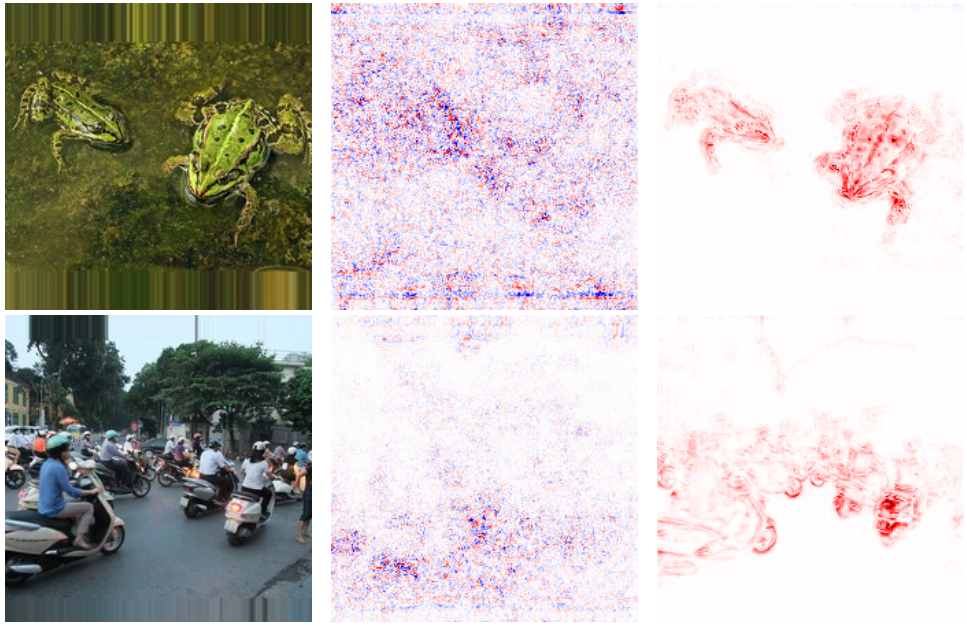


A more narrow scope: explaining predictions on a single sample by decomposition

- case of images: compute a score $r_d(x)$ for every input dimension d of the input sample $x = (x_1, \ldots, x_d, \ldots, x_D)$

$$\underbrace{f(x) \approx \sum_{d=1}^{D} r_d(x)}_{\textit{decomposition}}$$

- objective function is left open



What is a good explanation within the set of decomposition approaches?

What is a good explanation within the set of decomposition approaches?

- Is there a theoretically optimal approach?

Is there a theoretically optimal approach?

2. Shapley values

The Setup:

- Have function f, and a point to be explained $x = (x_1, \ldots, x_d)$.
- We can evaluate f on subsets $x_S = \{x_{i_1}, x_{i_2}, x_{i_3}, \ldots | \forall k : i_k \in S\}$ of features from x.

Shapley value

then the Shapley value is defined as:

$$\phi_j(f, x) = \frac{1}{d} \sum_{S \subseteq \{1,\ldots,d\} \setminus \{j\}} \frac{f(x_{S \cup \{j\}}) - f(x_S)}{\binom{d-1}{|S|}}$$

Its interpretation?

- Have function f, and a point to be explained $x = (x_1, \ldots, x_d)$.
- We can evaluate f on subsets $x_S = \{x_{i_1}, x_{i_2}, x_{i_3}, \ldots | \forall k : i_k \in S\}$ of features from x.
- then the Shapley value is defined as:

$$\phi_j(f, x) = \frac{1}{d} \sum_{S \subseteq \{1,\ldots,d\} \setminus \{j\}} \frac{f(x_{S \cup \{j\}}) - f(x_S)}{\binom{d-1}{|S|}}$$

$$= \frac{1}{\text{numsets}} \sum_{k \geq 1 \text{ sets } S \text{ without } j, |S|=k} \sum \frac{\text{differential contrib of } j \text{ to set } S}{\text{number of sets of same size } |S|}$$

Game Theory: S is set of players playing a game with outcome $f(x_S)$.
j a member which can join the set with outcome $f(x_{S \cup \{j\}})$

Favourable theoretical properties:

- does not make a difference, zero Shapley-value

$$\forall S: \; f(x_{S\cup\{j\}}) = f(x_S) \Rightarrow \phi_j(f,x) = 0$$

- value-equal pair of features j, k, identical Shapley-value:

$$\forall S: \; f(x_{S\cup\{j\}}) = f(x_{S\cup\{k\}}) \Rightarrow \phi_j(f,x) = \phi_k(f,x)$$

- Efficiency:

$$\sum_{j=1}^{d} \phi_j(f,x) = f(x) - E_X[f(X)]$$

  - decomposition of $f(x)$
  - distributes the difference between function value $f(x)$ and its expectation $E_X[f(X)]$ onto all dimensions in an equal way (not shown here[1])

[1] Grabisch https://www.worldscientific.com/doi/abs/10.1142/S0218488597000440

We have interpreted SHAP.
How to categorize this approach?

$$\phi_j(f,x) = \frac{1}{d} \sum_{S \subseteq \{1,\dots,d\} \setminus \{j\}} \frac{f(x_{S\cup\{j\}}) - f(x_S)}{\binom{d-1}{|S|}}$$

$$= \frac{1}{\text{numsets}} \sum_{k \geq 1} \sum_{\text{sets } S \text{ without } j, |S|=k} \frac{\text{differential contrib of } j \text{ to set } S}{\text{number of sets of same size } |S|}$$

We have interpreted SHAP.
How to categorize this approach?

$$\phi_j(f,x) = \frac{1}{d} \sum_{S \subseteq \{1,\ldots,d\} \setminus \{j\}} \frac{f(x_{S\cup\{j\}}) - f(x_S)}{\binom{d-1}{|S|}}$$

$$= \frac{1}{\text{numsets}} \sum_{k \geq 1 \text{ sets } S \text{ without } j, |S|=k} \sum \frac{\text{differential contrib of } j \text{ to set } S}{\text{number of sets of same size } |S|}$$

Combinatorially exhaustive occlusion differences for j
differential values: sets with player j, sets without j
(cf. Petliuk et al. RISE https://arxiv.org/abs/1806.07421)

- combinatorially exhaustive summing: can approximate by Monte-Carlo. Strumbelj, Kononenko (2014): https://link.springer.com/article/10.1007/s10115-013-0679-x

$$\phi_j(f, x) \approx \frac{1}{n} \sum_{l=1}^{n} f(x_{S_l \cup \{j\}}) - f(x_{S_l \setminus \{j\}})$$

- need it for a subset? Generalize it to a set notion: Grabisch+Roubens https://link.springer.com/article/10.1007%2Fs001820050125

We can evaluate f on subsets $x_S = \{x_{i_1}, x_{i_2}, x_{i_3}, \ldots | \forall k : i_k \in S\}$ of features from x. – Outside of tabular data types this is a very strong assumption.

- How to remove a region in an image, a language sentence ?
- How to remove an interval of a data point being a time series ?
- is S, $S \cup \{j\}$ plausible or outlier (bonds and molecules) ?

Dimensions of a data sample $x \leftrightarrow$ Players in a game ?

No optimality guarantee when

- applicability assumptions do not hold
- one has to use approximations (e.g. MC)

Published as a conference paper at ICLR 2019

L-SHAPLEY AND C-SHAPLEY: EFFICIENT MODEL INTERPRETATION FOR STRUCTURED DATA

Jianbo Chen[*] **Le Song**[†,§] **Martin J. Wainwright**[*,‡] **Michael I. Jordan**[*]
UC Berkeley[*], Georgia Institute of Technology[†], Ant Financial[§], Voleon Group[‡]

Chen, Song, Wainwright, Jordan *L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data*, *ICLR 2019*, https://openreview.net/forum?id=S1E3Ko09F7

Challenges of Shapley values in practice



Figure 10: The above figure shows explanation results on ten randomly selected figures of horses and deers. We mask 205 top pixels chosen by each method out of 1,024 pixels for each image. The first row shows the original images. The rest of the rows show images masked based on scores produced by ConnectedShapley, KernelSHAP and SampleShapley respectively.

Chen, Song, Wainwright, Jordan *L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data*, *ICLR 2019*, https://openreview.net/forum?id=S1E3Ko09F7

Yet: There is nothing wrong with Shapley values when the main assumption holds, and one can approximate them well.
Better looking results: https://github.com/slundberg/shap

What other methods exist for data types where Shapley assumptions do not hold well?

3. Linearizations

Starting point was: $\underbrace{f(x) \approx \sum_{i=1}^{d} r_d(x)}_{\text{Decomposition}}$

if linear model: $e(x) = w \cdot x = \sum_d w_d x_d$

$f(x) = \text{sign}(e(x))$

then use this: $r_d(x) = w_d x_d$

or this: Shapley for $e(x)$: $\phi_d(e, x) = w_d x_d - w_d E[x_d]$

- Starting point was: $f(x) = \sum_{i=1}^{d} r_d(x)$ f is non-linear now
- Taylor Expansion up to first order:

$$\begin{aligned} f(x) &\approx f(x_0) + \nabla f(x_0) \cdot (x - x_0) \\ &= f(x_0) + \sum_d \frac{\partial f}{\partial x_d}|_{x_0}(x_d - x_{0,d}) \end{aligned}$$

- use as explanation:

$$r_d(x) = \frac{\partial f}{\partial x_d}|_{x_0}(x_d - x_{0,d})$$

Linearizations

- Norms of Gradients (sensitivity)
- Gradient $\times$ Input
- Integrated Gradient
- LIME
- Grad-CAM

- have input sample $x = (x_1, \ldots, x_D)$, a prediction $f(x)$ (classification or any other setting)
- question: which parts of x are important for the prediction $f(x)$?

$$r_d(x) = \left(\frac{\partial f}{\partial x_d}|_x \right)^2$$

- An example when it is not optimally informative:

$$f(x) = w \cdot x \quad (1)$$

$$\text{Sensitivity: } \left(\frac{\partial f}{x_d}(x) \right)^2 = w_d^2 \quad (2)$$

$$\text{vs. SHAP: } r_d(x) = w_d x_d - E[w_d x_d] \quad (3)$$

  Explanation for a single score: w_d^2 ignores sign of input for classification.

- It is optimal in a sense: single-dimension sensitivities under small perturbations!

The main drawback:

What the gradient explains

- The gradient does **not** explain which pixels **are most contributing to the prediction** of a cat.
- The gradient explains which pixels **are most sensitive to change the prediction** of a cat.

Most sensitive to change locally $\neq$ most contributing

(+) easy to implement

(+) fast to compute

(−) often not the question you wanted to ask

Use as explanation:

$$r_d(x) = \frac{\partial f}{x_d}(x)\, x_d, \tag{4}$$

$$\mathbf{R} = (r_d, d = 1, \ldots, D) = \nabla f(x) \cdot x \tag{5}$$

⊙ derivation via global Taylor decomposition for a point x_0 orthogonal to the gradient ($\nabla f(x) \cdot x_0 = 0$) in the point x to be explained.

$$f(x_0) = f(x) + \nabla f(x) \cdot (x_0 - x) + \mathcal{O}(\|x - x_0\|^2) \tag{6}$$

$$\Rightarrow f(x) \approx f(x_0) + \nabla f(x) \cdot (x - x_0) \tag{7}$$

$$= f(x_0) + \nabla f(x) \cdot x - \underbrace{\nabla f(x) \cdot x_0}_{=0} \tag{8}$$

$f(x_0)$ is a scalar term, independent of any dimension

(−) can be noisy for deep ReLU-networks due to gradient shattering

Gradient shattering

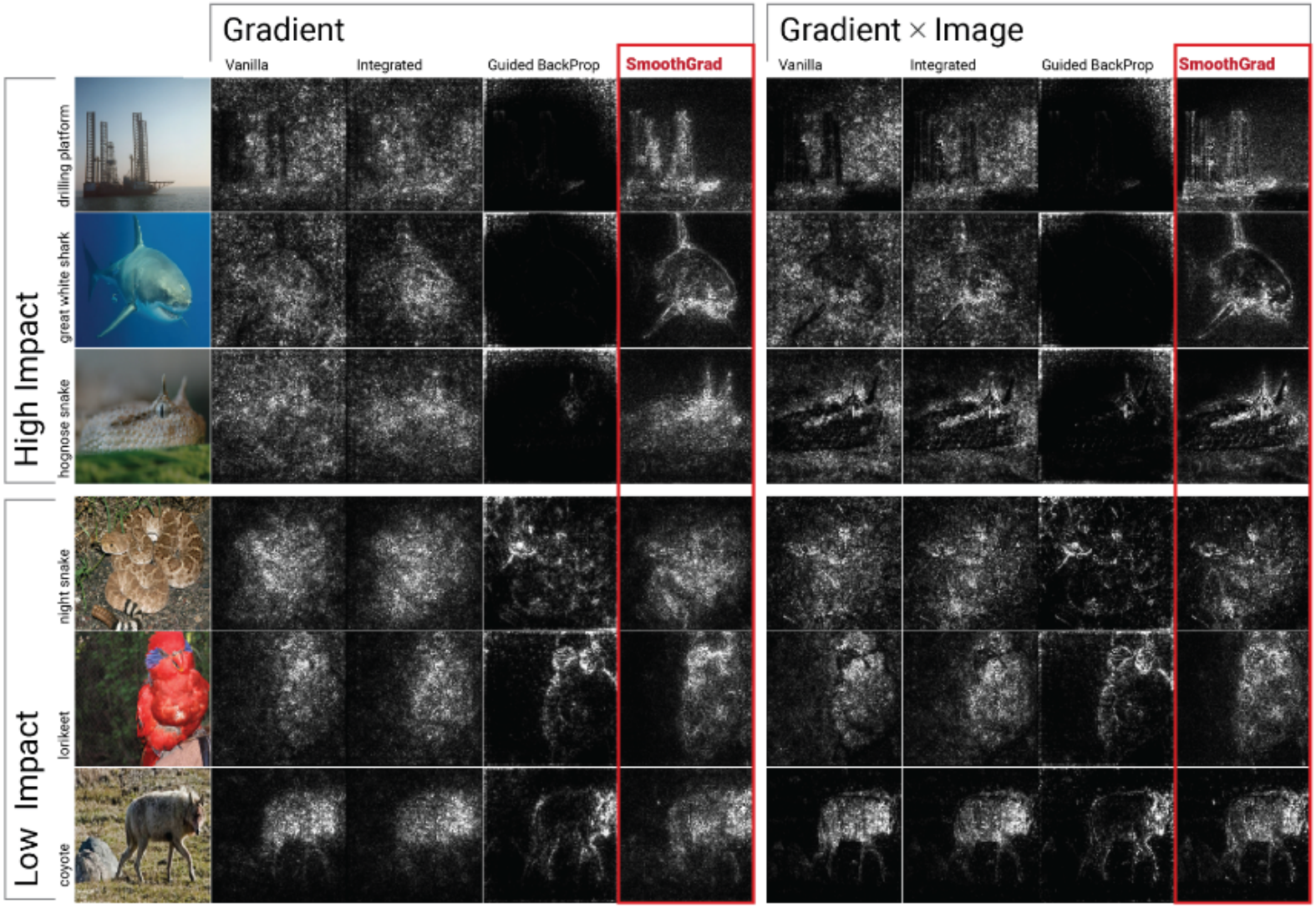


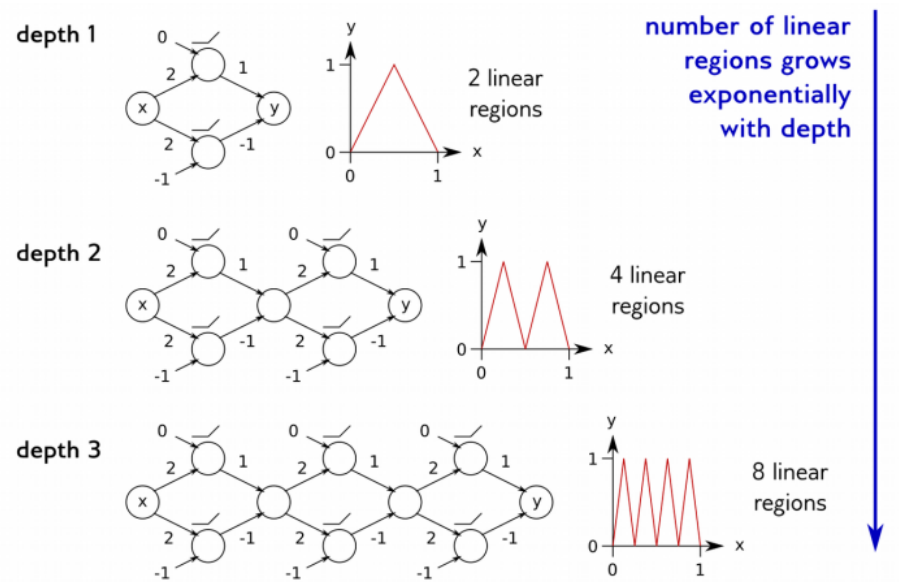
Figure 5. Qualitative evaluation of different methods. First three (last three) rows show examples where applying SMOOTHGRAD had high (low) impact on the quality of sensitivity map.

Credit: https://arxiv.org/pdf/1706.03825.pdf

The noiseness of gradient × input and related methods for deep ReLU-networks: Gradient Shattering Effect

- Montufar et al. 2014 https://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf.
- Balduzzi et al. 2017 http://proceedings.mlr.press/v70/balduzzi17b/balduzzi17b.pdf.

The Shattered gradients problem [Montufar'14, Balduzzi'17]



Worth reading: works of Marco Ancona

Integrated Gradient: *Axiomatic attribution for deep networks*
Sundararajan et al., *ICML 2017*
a heuristic very similar to the gradient times input idea:

$$R_d(x) = (x_d - x_d^{(0)}) \frac{1}{m} \sum_{k=1}^{m} \frac{\partial f}{\partial x_d} |_{z=x^{(0)}+\frac{k}{m}(x-x^{(0)})}$$

- $\odot$ Averages over partial derivatives along multiple points $x^{(0)} + \frac{r}{R}(x - x^{(0)})$ along a path from $x^{(0)}$ to x.
- $(+)$ Why that can be better than gradient $\times$ input? Averaging gradients smoothes out the gradient shattering.
- $(+)(-)$ IG gets better and slower when hundreds of points are used ($+$ slows down).

Smilkov et al. https://arxiv.org/pdf/1706.03825.pdf
Take the averaging idea further

- if you have any attribution map method $M(x)$ which depends on an input image x, then:
- compute an average over many slightly noised $x + \mathcal{N}(0, \sigma^2)$ input images

$$\widehat{M}(x) = \frac{1}{K} \sum_{i=1}^{K} M(x + n_i), n_i \sim \mathcal{N}(0, \sigma^2)$$

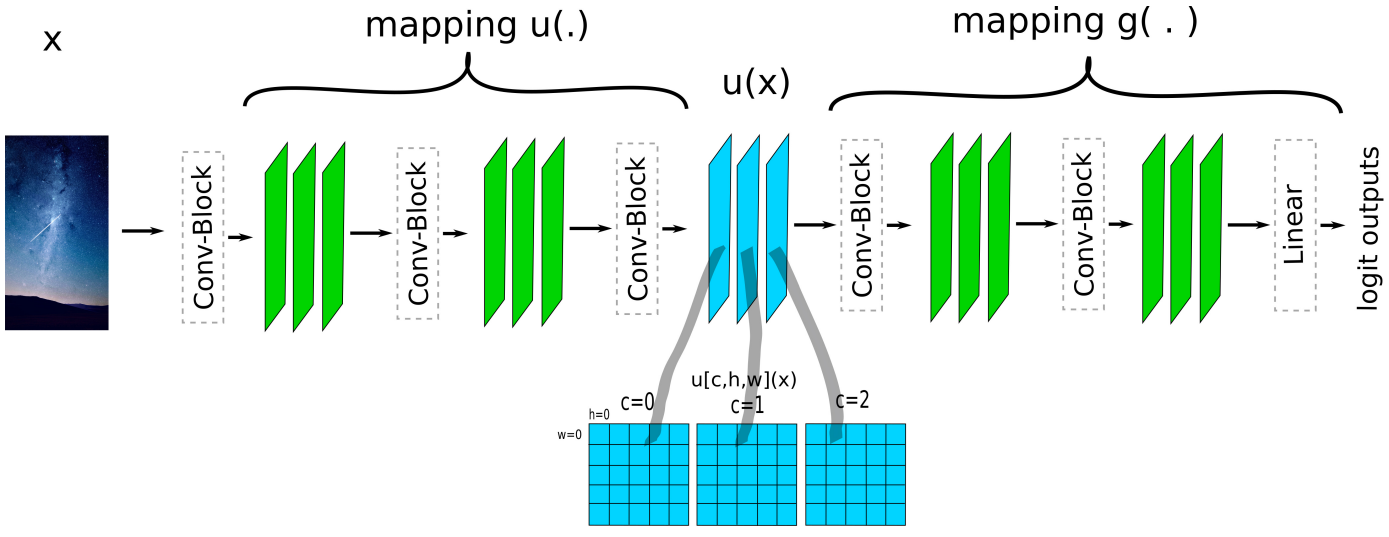
Compare to integrated gradient:

$$R_d(x) = \frac{1}{K} \sum_{i=1}^{K} (x_d - x_d^{(0)}) \frac{\partial f}{\partial x_d}|_{z=x^{(0)}+\frac{i}{K}(x-x^{(0)})}$$

IG samples along a line from $x^{(0)}$ to x (and uses a special $M(x)$).
Smoothgrad samples from a normal distribution around x.

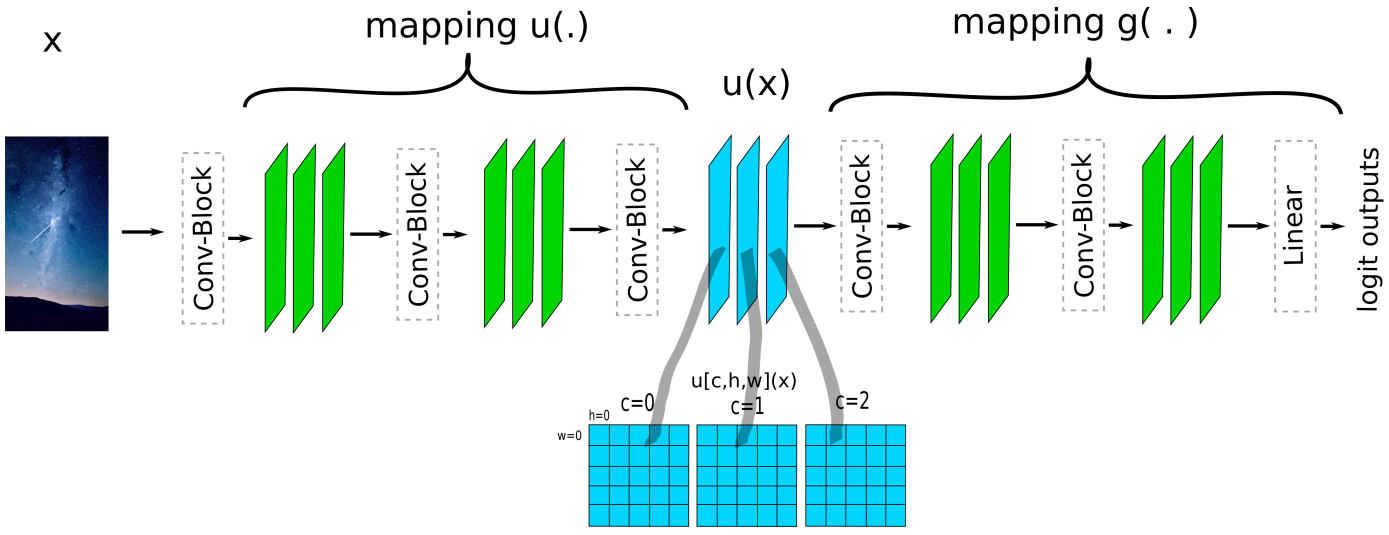
A heuristic variation on the gradient times input idea - applied in feature map space. https://arxiv.org/abs/1610.02391

- given a feature map $u(x)$ with components $u[c, h, w](x)$ having C channels, width W, height H, and a predictor $f(x) = g(u(x))$ where $g(\cdot)$ are some layers on top of u.



Gradient times input in the space of u would be:

$$R[c, h, w] = \nabla^u \, g|_{u(x)} \odot u(x) = \left. \frac{\partial g}{\partial u[c, h, w]} \right|_{u(x)} u[c, h, w](x)$$

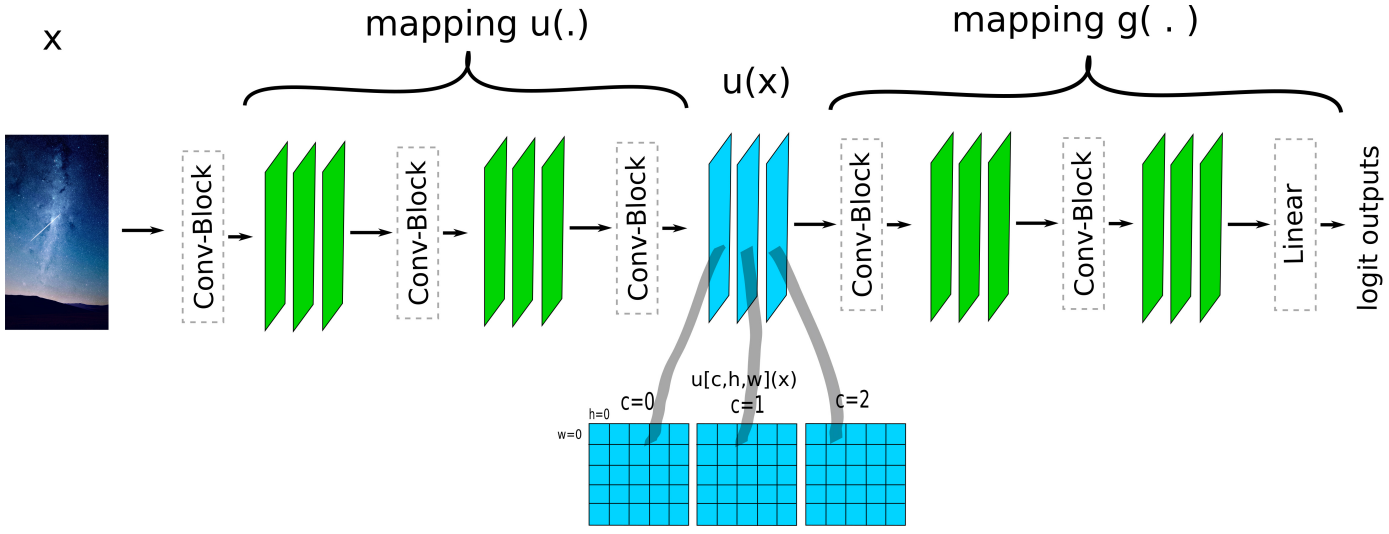


Gradient times input in the space of u would be:

$$R[c,h,w] = \nabla^u \, g|_{u(x)} \odot u(x) = \left.\frac{\partial g}{\partial u[c,h,w]}\right|_{u(x)} u[c,h,w](x)$$

Grad-Cam replaces the partial derivative $\frac{\partial g}{\partial u[c,h,w]}$ by a spatially averaged version

$$\alpha_c = \frac{1}{HW} \sum_{h,w} \left.\frac{\partial g}{\partial u[c,h,w]}\right|_{u(x)}$$



Grad-Cam replaces the partial derivative $\frac{\partial g}{\partial u[c,h,w]}$ by a spatially averaged version

$$\alpha_c = \frac{1}{HW} \sum_{h,w} \frac{\partial g}{\partial u[c,h,w]}\Big|_{u(x)}$$

$$R[h,w] = ReLU(\sum_{c=1}^{C} \alpha_c \; u[c,h,w])$$

and applies a weighted sum $\sum_c$ over all channels in this layer+ a *ReLU*

Grad-Cam replaces the partial derivative $\frac{\partial g}{\partial u[c,h,w]}$ by a spatially averaged version

$$\alpha_c = \frac{1}{HW} \sum_{h,w} \frac{\partial g}{\partial u[c,h,w]} \Big|_{u(x)}$$

$$R[h,w] = ReLU(\sum_{c=1}^{C} \alpha_c \; u[c,h,w])$$

high level view:

- gradient times input in feature space of a chosen layer
- with spatial averaging of gradients

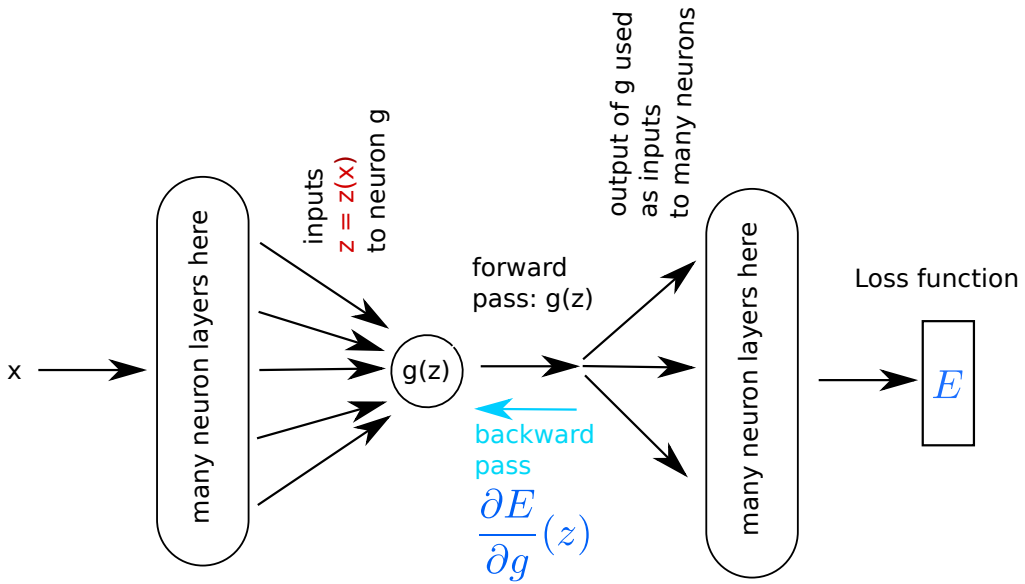
- ⊙ the spatial averaging helps to smooth with gradient shattering
- ⊙ the relu suppresses spatial locations with approximately irrelevant contributions to the score
- ⊙ need to choose a feature map as free parameter, in principle extendible to an overlay of responses of multiple layers (why choose only a single layer?)

(+) simple to implement

(−) low resolution unless one does pixel-wise multiplication with guided backprop (but then why not using guided BP right away?)

Backpropagation with a heuristic to cancel out parts of the backpropagated scores:
Consider a relu activation neuron $g(z) = relu(w \cdot z + b)$:

- this receives in the forward pass the value vector $z = z(x)$ from the layers below.
- In the backward pass it received the derivative with respect to itself $\frac{\partial E}{\partial g}(z)$ from the layers above.



Guided backprop says: do backprop but **zero out incoming gradient** $\frac{\partial E}{\partial g}(z)$ at an activation layer ... when passing to through $g()$ if:

- the input to the activation of the neuron are negative $z < 0$
- the gradient arriving at this neuron is negative $\frac{\partial E}{\partial g}(z) < 0$

Why? Look only on the bright side of life

- if the input to the activation of the neuron is negative $z < 0$, then z is a suppressing neuron. rule: Ignore gradients from suppressing neurons, pass through only gradient signal from activating neurons
- if the gradient arriving at this neuron is negative. rule: ignore gradients which decrease the function value, look only at gradients which increase the prediction
- its a heuristic to look only at one end of effects: activating signals and gradients

Need to take the absolute value to get something useful. Sign has no meaning in there.

(+) gives often clean heatmaps, high resolution heatmaps

(+) very easy to implement using a backward hooks at ReLUs

(–) its a heuristic, no theoretical underpinning. Not really sure what it does.

(–) open what to do with non-relu activations

(–) high precision but lack of sensitivity to explanation for different classes

- take an input $x = (x_1, \ldots, x_D)$. Occlude a subset $S \subset \{x_1, \ldots, x_D\}$ of dimensions by some value $x \mapsto \tilde{\tilde{x}}$.
- measure $r_S(x) = f(x) - f(\tilde{\tilde{x}})$ as sensitivity of the subset S
- order subsets S according $r_S(x)$ to find the most sensitive subsets

Choice of subset scale matters! Example

- simplest case: subset consists of single dimension $S = \{x_d\}$
- using subsets of more than one element allows to capture correlations in f, example

$$f(x) = \max(|x_1|, |x_2|) + |x_3|$$

only changing x_1 to a value $x_1 = 0$ wont change $f(x)$, thus would mask sensitivity of f to inputs in x_1, x_2.

- images: what pixel size to occlude ? 5×5 ? or superpixels from a superpixel segmentation algorithm? or extended superpixels to cover boundaries?

need to have a clear idea how to **occlude a region**

- can be simple for certain financial tasks. example: replace an input dimension value by the median, or measure for (median, 25%-,75%-quantile) and choose the largest sensitivity over quantiles
- images – can get more complicated ... black square ? from another image ? gray square?
- uncertainty? $\longrightarrow$ compute statistic e.g. median over multiple occlusions of the same region if there is no good unique occlusion
- images: can use GAN inpainting (deepfill) to generate occlusions which do not result in outliers
  https://openaccess.thecvf.com/content/ACCV2020/html/Agarwal_Explaining_image_classifiers_by_removing_input_features_using_generative_models_ACCV_2020_paper.html

(+) direct measurement of local sensitivity

(+) super simple to implement: create n copies of an image, do mod in each copy, run forward pass

(+) forward pass only, black-box compatible

(–) need to decide on occlusion region size and how to occlude

(–) slow, really

Starting point of many methods:

- given: A. trained model f, B. a prediction $f(x)$ for input $x = (x_1, \ldots, x_d, \ldots, x_D)$.
- general case: To compute a relevance score $r_d(x)$ for every input dimension x_d of input x explaining the prediction $f(x)$, such that approximately:

$$f(x) \approx \sum_{d=1}^{D} r_d(x) \quad \leftarrow \text{ decomposition with constraints} \tag{9}$$

- one way to achieve this, is first order Taylor decomposition around some point $x^{(0)}$

$$f(x) \approx f(x^{(0)}) + \nabla f|_{x_0} \cdot (x - x^{(0)}) = \sum_d \left.\frac{\partial f}{\partial x_d}\right|_{x_0} (x_d - x_d^{(0)}) = \sum_d r_d(x)$$

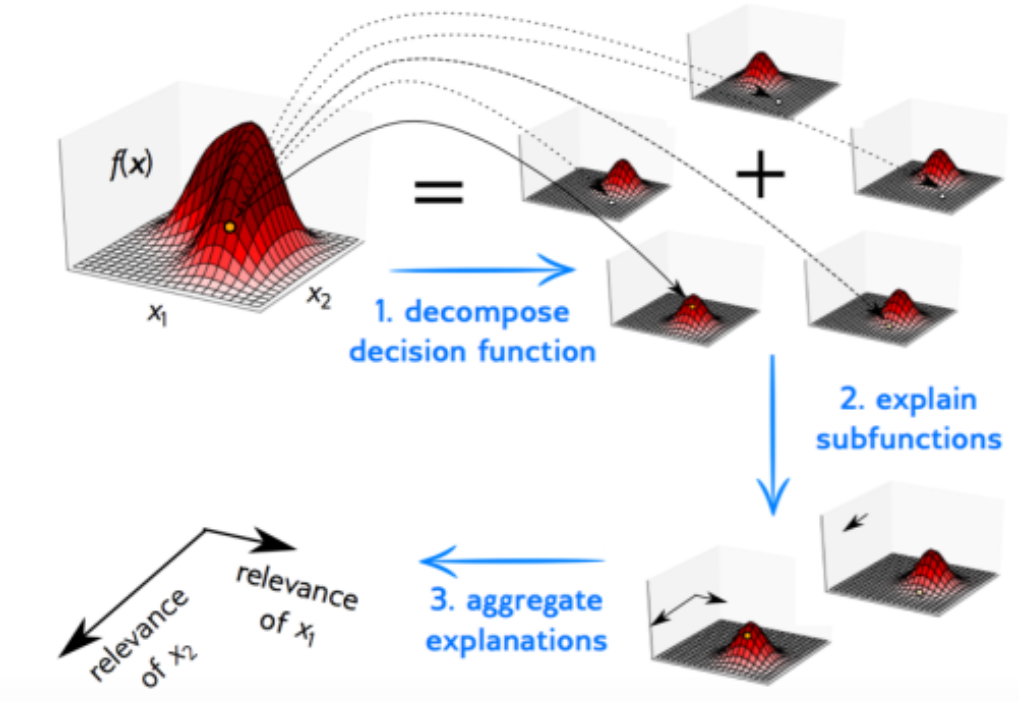
$$r_d(x) := \left.\frac{\partial f}{\partial x_d}\right|_{x_0} (x_d - x_d^{(0)})$$

- linearizes relatively to $x^{(0)}$. Explains the difference of prediction in x relative to the prediction made in $x^{(0)}$. Note: a relative explanation!
- the next method applies such an idea for every layer, and stacks decompositions

image LRP-α-β

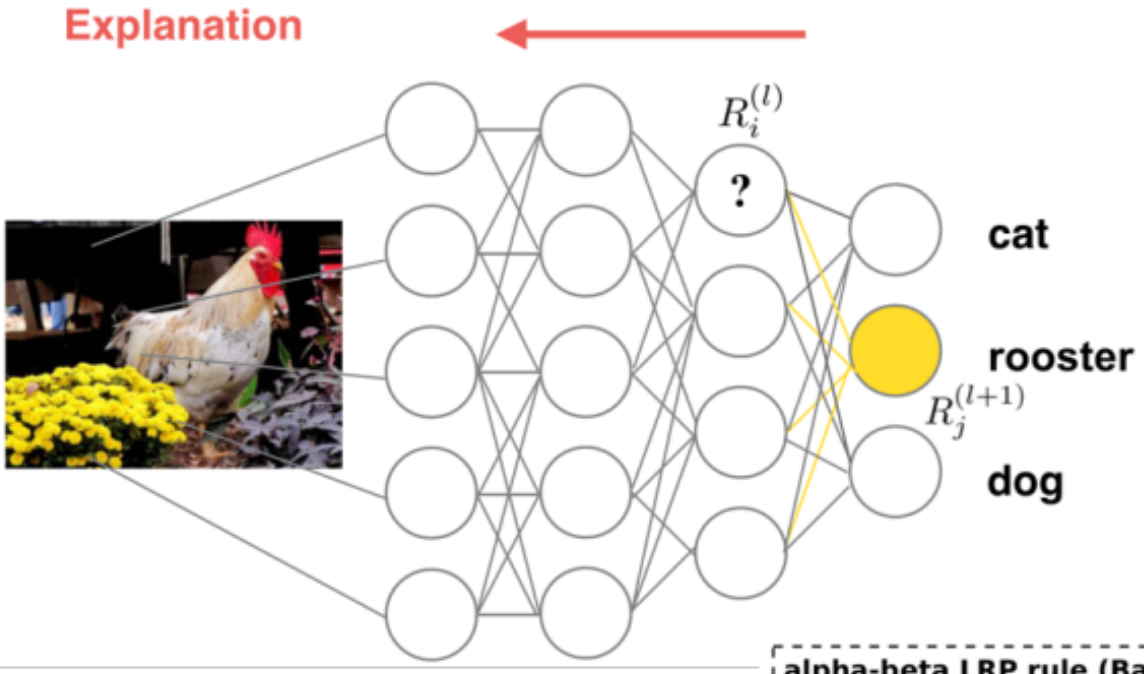


- Divide and conquer: decompose network in layers



credit: W. Samek

- Taylor approximation per layer/neuron
- easier to find roots for one layer
- robustness to gradient shattering



Theoretical interpretation
Deep Taylor Decomposition
(Montavon et al., 2017)

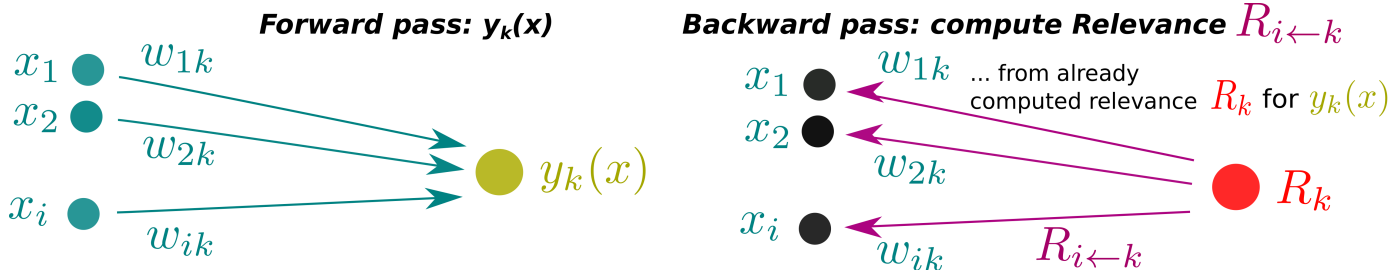
alpha-beta LRP rule (Bach et al. 2015)

$$R_i^{(l)} = \sum_j (\alpha \cdot \frac{(x_i \cdot w_{ij})^+}{\sum_{i'} (x_{i'} \cdot w_{i'j})^+} + \beta \cdot \frac{(x_i \cdot w_{ij})^-}{\sum_{i'} (x_{i'} \cdot w_{i'j})^-}) R_j^{(l+1)}$$

where $\alpha + \beta = 1$

credit: W. Samek

The next 5 slides and the slide which root for which layer you do not need to memorize.



$$\text{forward pass: } y_k = g\left(\sum_i w_{ik}x_i + b\right) \tag{10}$$

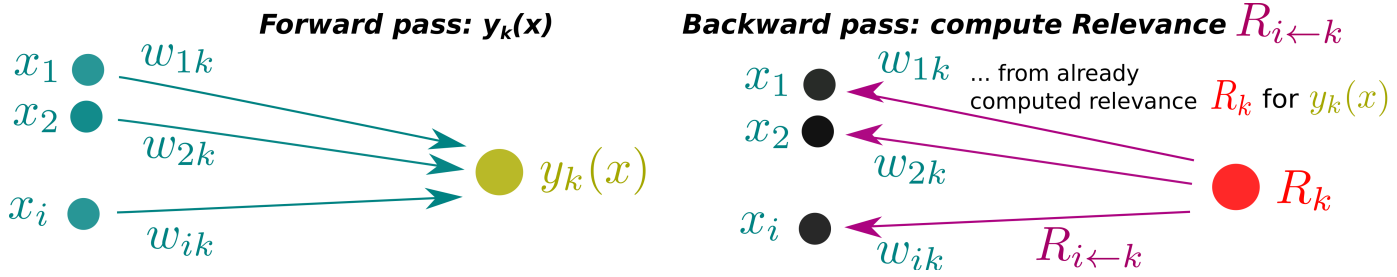
backward: have computed already the relevance R_k for the neuron output y_k (11)

$$\text{LRP backward: } R_{i\leftarrow k}(\mathbf{x}) = R_k M_{i\leftarrow k}(w,x)$$

fraction of positive part of input dim x_i relative to all inputs $x_{i'}$

$$\beta = 0: \quad M_{i\leftarrow k} = \overbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}} \tag{12}$$

$$\beta > 0: M_{i\leftarrow k} = (1+\beta)\underbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}}_{\text{positive contributions}} - \beta\underbrace{\frac{(w_{ik}x_i)_-}{\sum_{i'}(w_{i'k}x_{i'})_-}}_{\text{negative contributions}} \tag{13}$$



$$\text{forward pass: } y_k = g\left(\sum_i w_{ik}x_i + b\right) \tag{10}$$

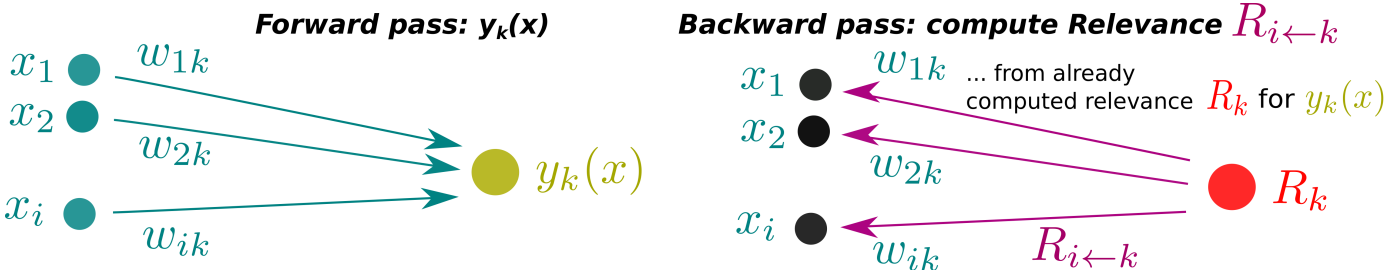
backward: have computed already the relevance R_k for the neuron output y_k (11)

$$\text{LRP backward: } R_{i\leftarrow k}(\mathbf{x}) = R_k M_{i\leftarrow k}(w,x)$$

fraction of positive part of input dim x_i relative to all inputs $x_{i'}$

$$\beta = 0: \quad M_{i\leftarrow k} = \overbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}} \tag{12}$$

$$\beta > 0: M_{i\leftarrow k} = (1+\beta)\underbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}}_{\text{positive contributions}} - \beta\underbrace{\frac{(w_{ik}x_i)_-}{\sum_{i'}(w_{i'k}x_{i'})_-}}_{\text{negative contributions}} \tag{13}$$



$$\text{forward pass: } y_k = g\left(\sum_i w_{ik}x_i + b\right) \tag{10}$$

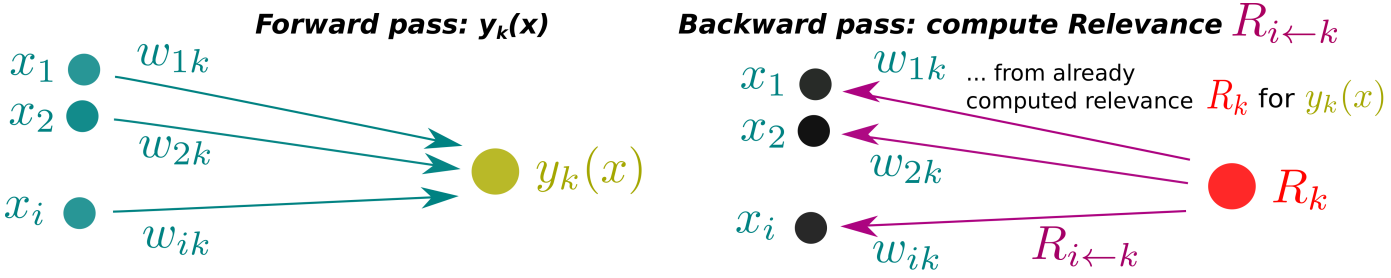
backward: have computed already the relevance R_k for the neuron output y_k (11)

$$\text{LRP backward: } R_{i\leftarrow k}(\mathbf{x}) = R_k M_{i\leftarrow k}(w,x)$$

fraction of positive part of input dim x_i relative to all inputs $x_{i'}$

$$\beta = 0: \quad M_{i\leftarrow k} = \overbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}} \tag{12}$$

$$\beta > 0: M_{i\leftarrow k} = (1+\beta)\underbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}}_{\text{positive contributions}} - \beta\underbrace{\frac{(w_{ik}x_i)_-}{\sum_{i'}(w_{i'k}x_{i'})_-}}_{\text{negative contributions}} \tag{13}$$



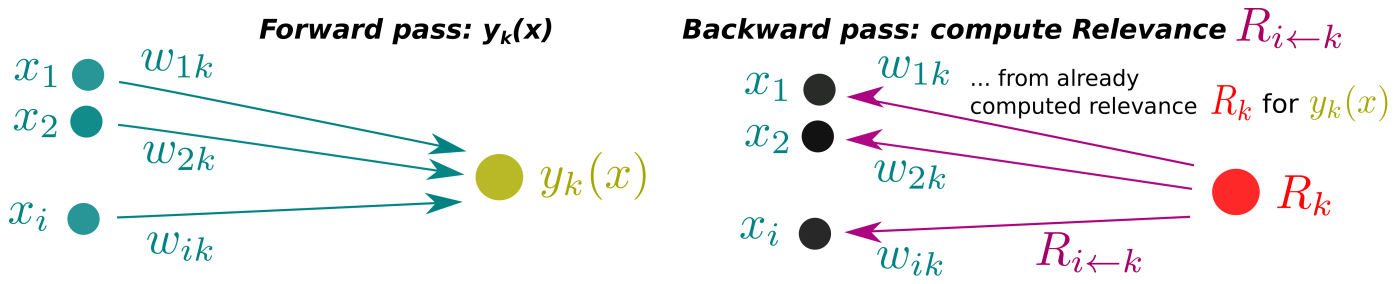
given: have computed already R_k as relevance of neuron output

$$y_k = g(\sum_i w_{ik} x_i + b) \tag{14}$$

$$R_{i\leftarrow k}(\mathbf{x}) = R_k M_{i\leftarrow k}(w, x)$$

$$M_{i\leftarrow k} = (1+\beta) \underbrace{\frac{(w_{ik}x_i)_+}{\sum_{i'}(w_{i'k}x_{i'})_+}}_{\text{positive contributions}} - \beta \underbrace{\frac{(w_{ik}x_i)_-}{\sum_{i'}(w_{i'k}x_{i'})_-}}_{\text{negative contributions}} \tag{15}$$

- β controls ratio of negative to total relevance $= \frac{\beta}{1+2\beta}$.
- negative to total relevance – fixed independent of neuron inputs x_i (!).
- bounded relevance scale: $|R_{i\leftarrow k}| \leq (1+\beta)|R_k|$ (not true for LRP-ϵ!)



Got $R_{i\leftarrow k}$ from R_k. How to compute R_i ?

$$R_i := \sum_{k:\, i \text{ is input to } k} R_{i\leftarrow k} \tag{16}$$

Same as in backpropagation

which root for which layer?

Name	Formula	layers
LRP-ϵ	$\sum_k R_k \left(\frac{x_i w_{ik}}{\sum_i x_i w_{ik} + b + \epsilon \operatorname{sign}(z)} \right)$	Linear
LRP-$\beta = 0$	$\sum_k R_k \left(\frac{(x_i w_{ik})_+}{\sum_i (x_i w_{ik})_+} \right)$	conv
LRP-γ	$\sum_k R_k \left(\frac{\gamma (x_i w_{ik})_+ + (x_i w_{ik})}{\sum_i \gamma (x_i w_{ik})_+ + \gamma (b)_+ + \sum_i (x_i w_{ik}) + b} \right)$	conv
LRP-z_β	$\sum_k R_k \left(\frac{x_i w_{ik} - l_i (w_{ij})_+ + h_i (w_{ij})_-}{\sum_i x_i w_{ik} + b - l_i (w_{ij})_+ + h_i (w_{ij})_-} \right)$	first conv layer
LRP-w^2	$\sum_k R_k \frac{w_{ik}^2}{\sum_i w_{ik}^2}$	same

Its a mere serving suggestion. Define a loss and measure the quality of your explanations and choose by that!

start memorizing here again :)

(–) Technically one root per layer. Need to validate choices by measuring explanation quality.

(–) using everywhere LRP-ϵ reduces to $\nabla f(x) \odot x$

(–) poor recall due to sparsity, gradient based methods often have better recall.

(+) high precision, gradient based methods often have lower precision

(+) works well for images with established presets (ignore biases, use LRP-β, γ), LSTMs, high-dimensional data

(+) does reasonably well when properly implemented, e.g.

Evaluating Explanation Methods for Deep Learning in Security, Warnecke, Arp, Wressnegger, Rieck, *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*

how to evaluate the quality of an attribution map?

- correlation to human intuition
- correlation to bounding boxes
- correlation to other ground truth? Problem with that: a neural network might learn other correlations present in your data(!!!) than your ground truth!!!
- important precision versus recall measures can differ a lot among methods
  - gradient-based: lower precision, higher recall (due to noise)
  - lrp/guided BP/deep LIFT: higher precision, lower recall (due to sparsity)
- authors opinion: there is not the one ...

stop memorizing here again :)

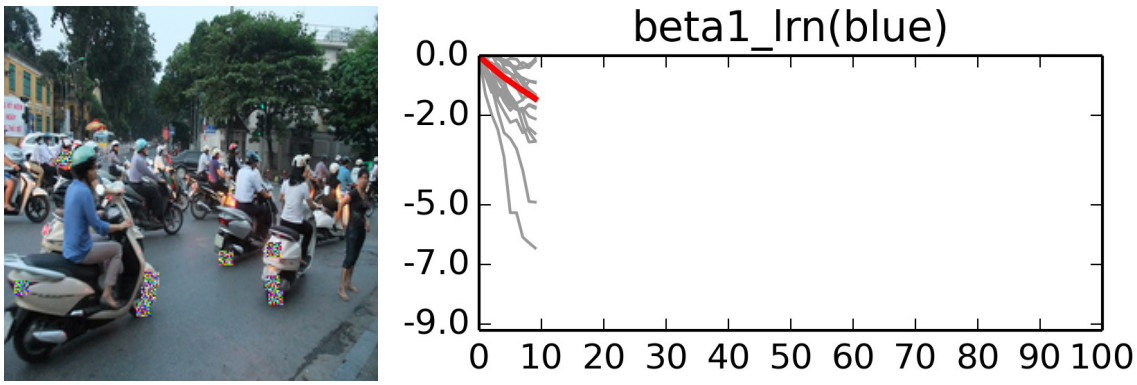
how to evaluate the quality?

One approach (not perfect): measure correlation to predictor sensitivity under change of regions.
https://ieeexplore.ieee.org/abstract/document/7552539,
https://arxiv.org/abs/1509.06321 (disclaimer: own work)

- choose region size, compute average heatmap score for each region
- Idea: use relevance scores (from any method) r to order regions.
- *modify* regions in the implied order
- measure decrease of prediction $f(\mathbf{x})$ under modifications
- key idea: if most important pixels get the highest score, then prediction will decrease fast, as more and more pixels get modified



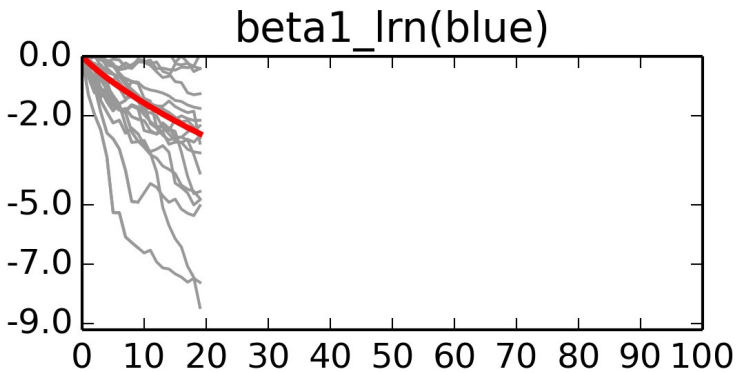
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

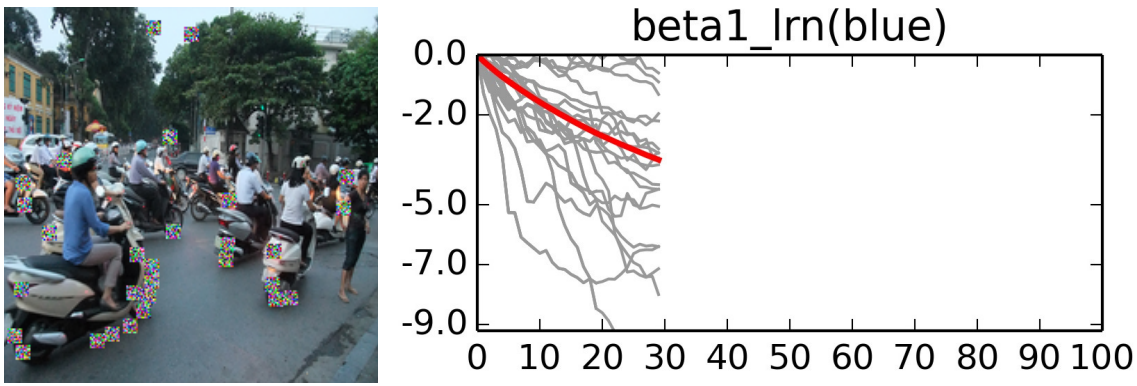
- Idea: measure area under graph - averaged over many runs and images (red line)





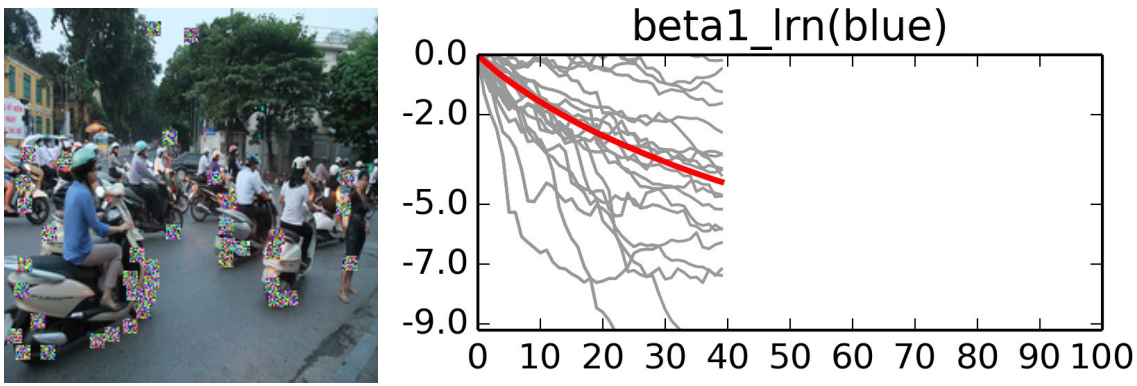
- can compare against random orderings of pixels

- Idea: measure area under graph - averaged over many runs and images (red line)



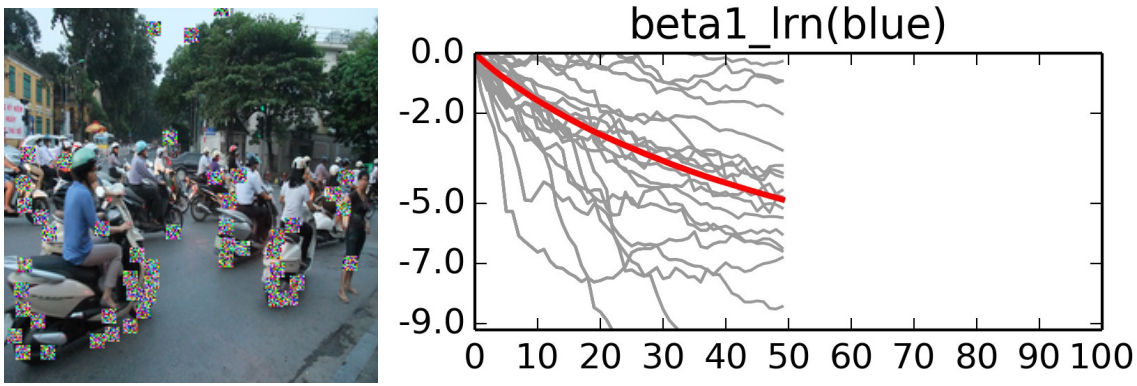
- can compare against random orderings of pixels

- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

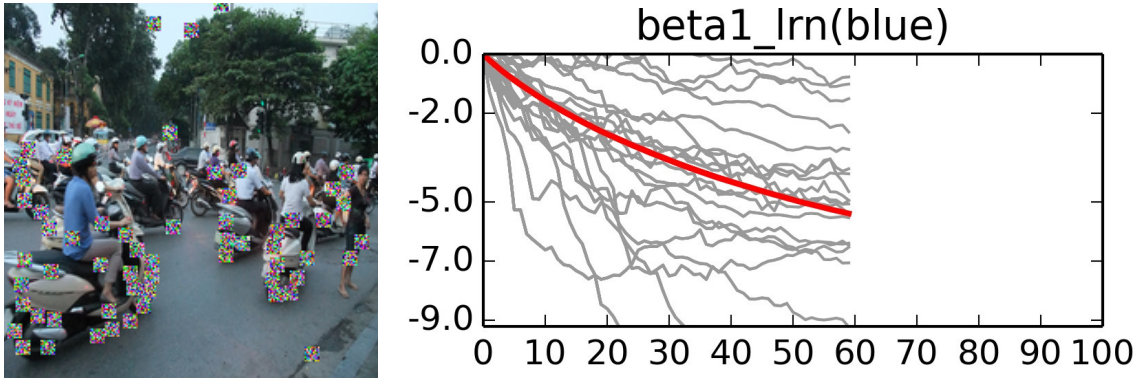
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

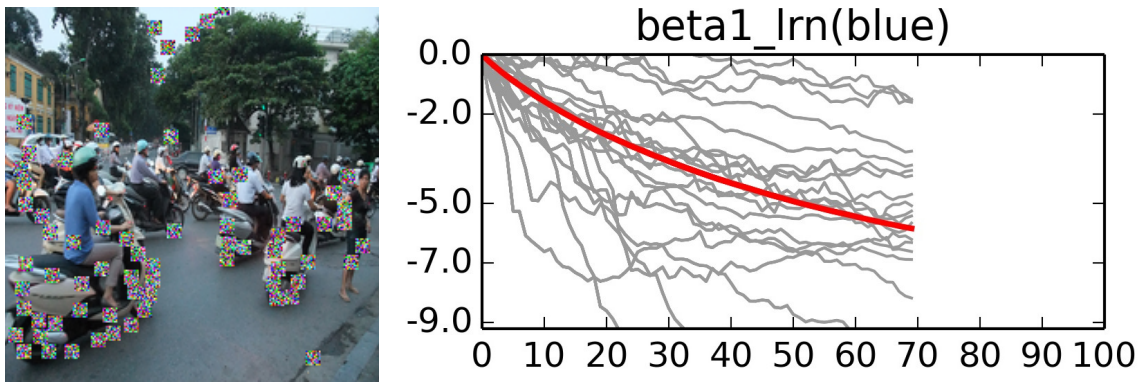
Evaluation?

- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

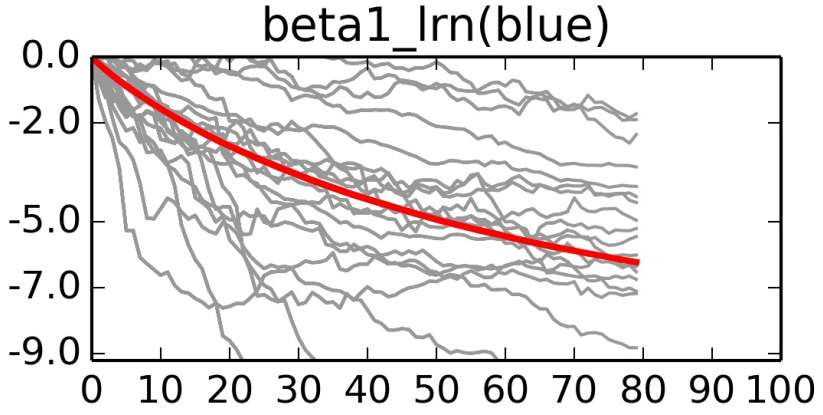
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

- Idea: measure area under graph - averaged over many runs and images (red line)

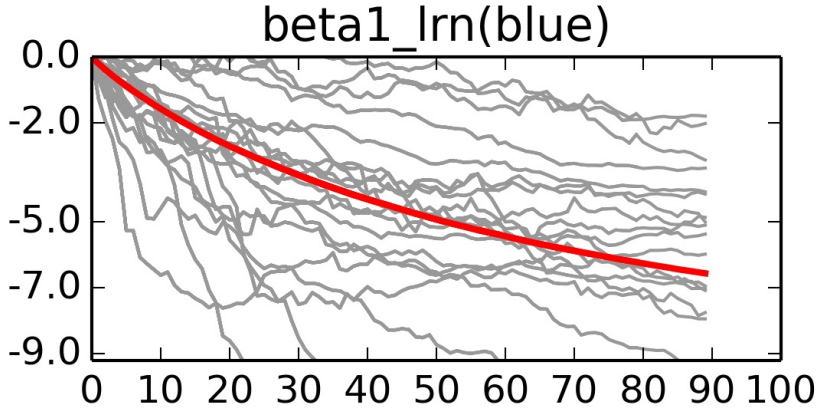




- can compare against random orderings of pixels

- Idea: measure area under graph - averaged over many runs and images (red line)





- can compare against random orderings of pixels

In practice,

- compute average decrease for many different modifications of the same region
- evaluation does not need image labels



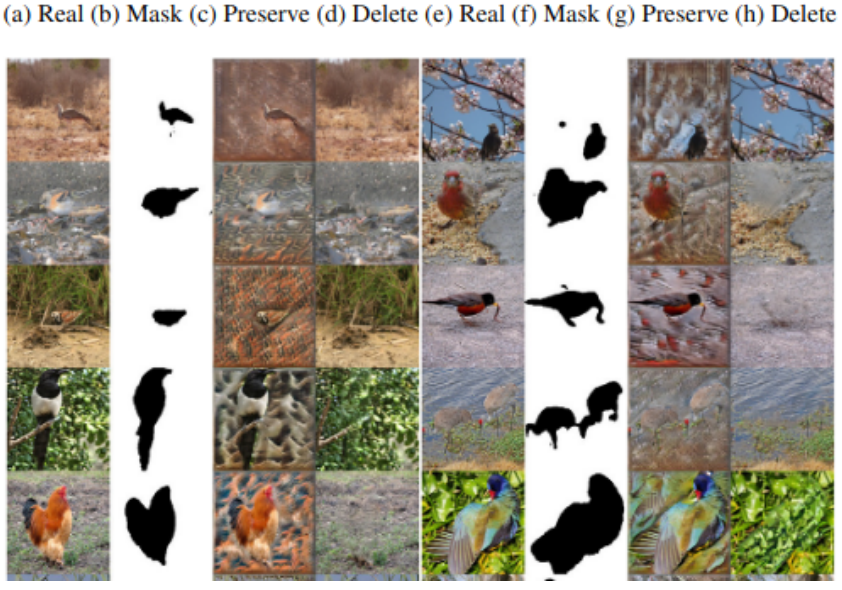
Evaluating the visualization of what a Deep Neural Network has learned, Samek et al., IEEE TNNLS 2017

https://ieeexplore.ieee.org/abstract/document/7552539,
https://arxiv.org/abs/1509.06321 (disclaimer: own work)

- choose region size
- compute average heatmap score per region
- sort regions descendingly
- sequentially perturb each region according to sorting oder
- measure sensitivity under sequential occlusion
- downside: occlusion as above creates outliers!!! Experiments were performed in 2015. Today: use GAN-inpainting or similar (see next slide).

- Measuring explanation quality wrt outliers as baseline is problematic. What is a meaningful inlier modification in an image?

  Explaining image classifiers by removing input features using generative models, Agarwal & Nguyen, *ACCV 2020*
  https://openaccess.thecvf.com/content/ACCV2020/papers/Agarwal_Explaining_image_classifiers_by_removing_input_features_using_generative_models_ACCV_2020_paper.pdf



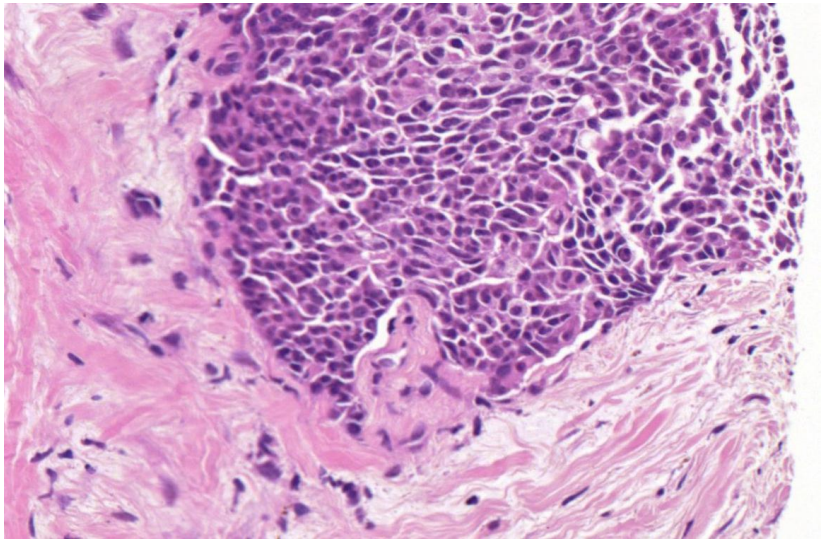
start memorizing here again :)

A. Iterative Dataset Design (medical imaging): Identify what you need to label for the next round

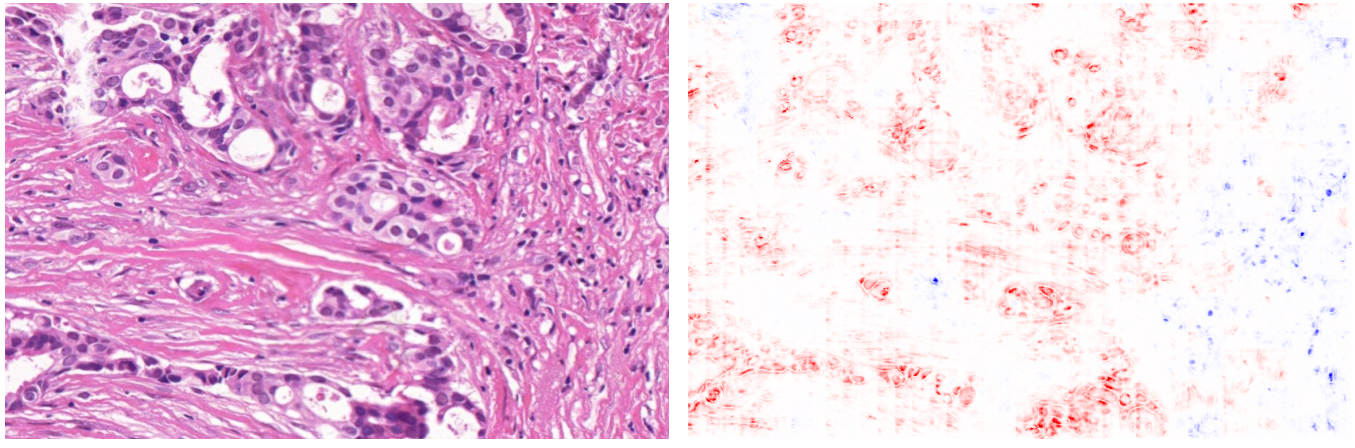
B. improving model performance in small-sample size tasks: LRP-guided training to improve cross-domain few shot learning

Case A: Iterative Dataset Design (medical imaging):
Identify what you need to label for the next round

- some problems: labels very costly, unlabeled data abundant
- biochemistry etc

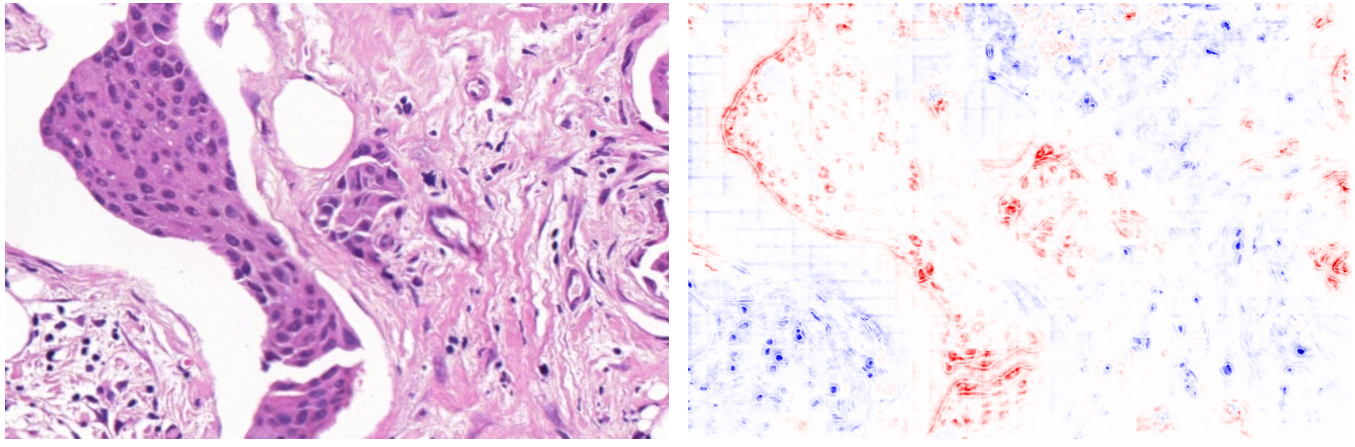


- improve model via growing the dataset
- decide what unlabeled data to add into next iteration of train and test set



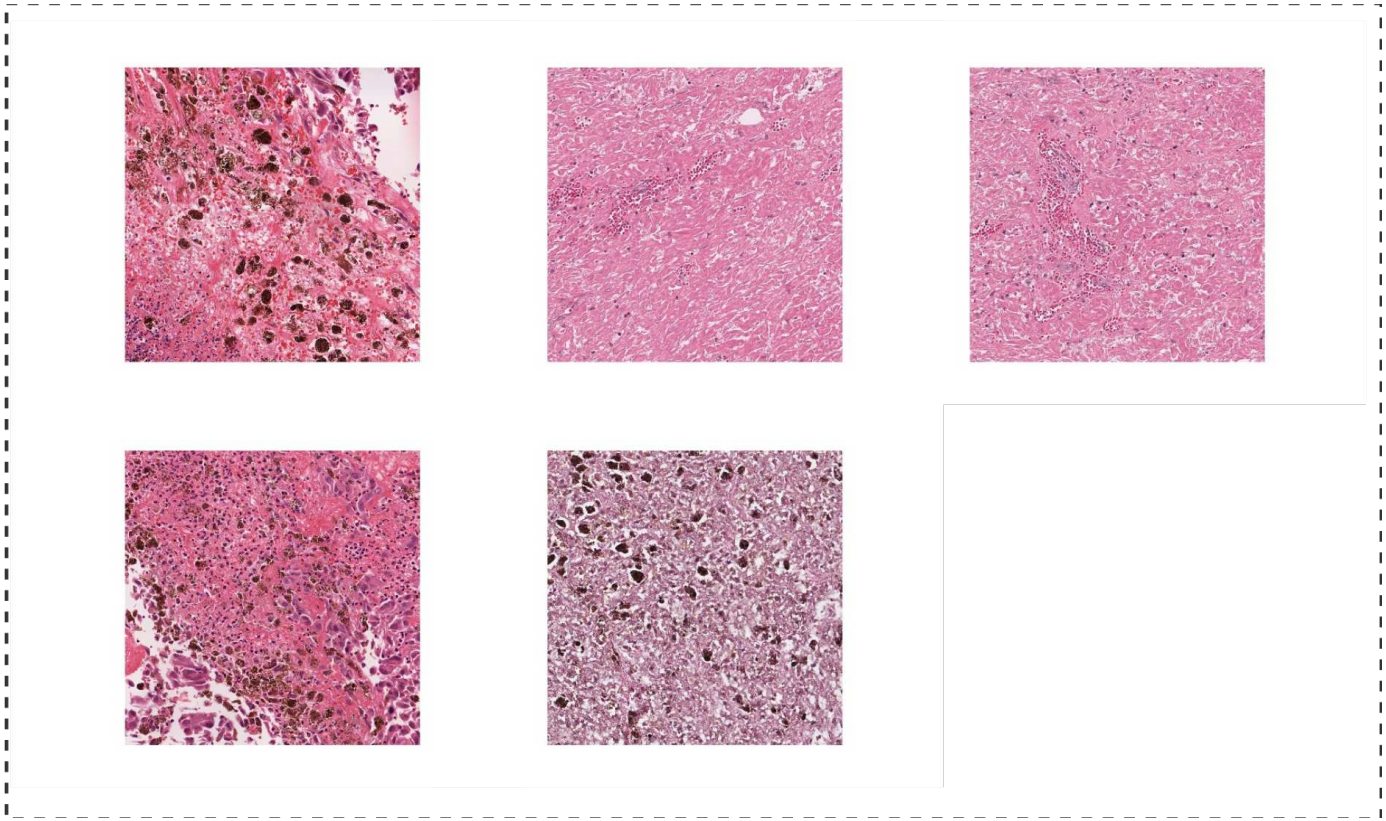
- Interpretability for efficiency in the selection step before labelling!

- improve model via growing the dataset
- decide what unlabeled data to add into next iteration of train and test set – precursor to labelling.



- Interpretability for efficiency in the selection step before labelling!

Original HE images with necrosis regions:

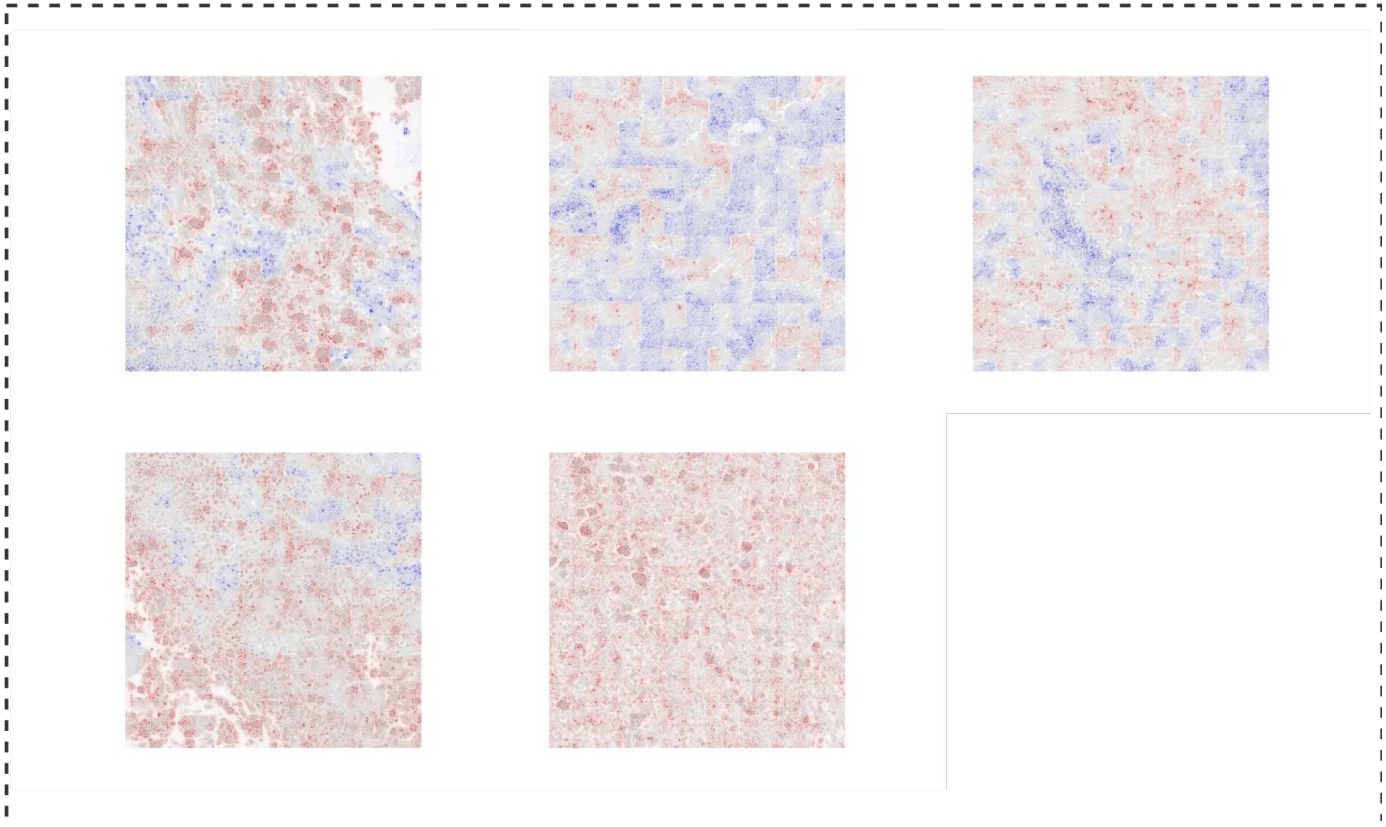


Hägele, Seegerer, Lapuschkin, Bockmayr, Samek, Klauschen, Müller, Binder, Resolving challenges in deep learning-based analyses of histopathological images using explanation methods,

Nat Sci Rep 2020

Iterative Dataset Design (Medical Data): what to label?

Finding unlabeled subclasses: Training **without** labeled necrosis samples.

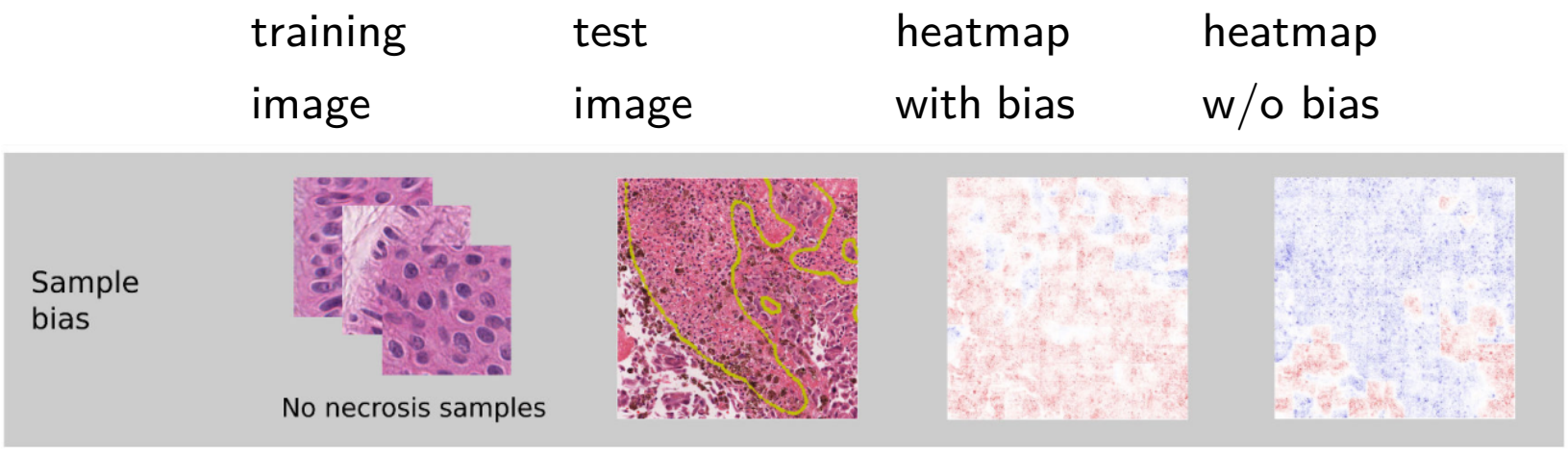


Hägele, Seegerer, Lapuschkin, Bockmayr, Samek, Klauschen, Müller, Binder, Resolving challenges in deep learning-based analyses of histopathological images using explanation methods,

Nat Sci Rep 2020

Iterative Dataset Design (Medical Data): what to label?

- left heatmap: false positive scores on unlabeled subclass.
- right heatmap: after augmenting training dataset with necrosis samples (labeled as negative)

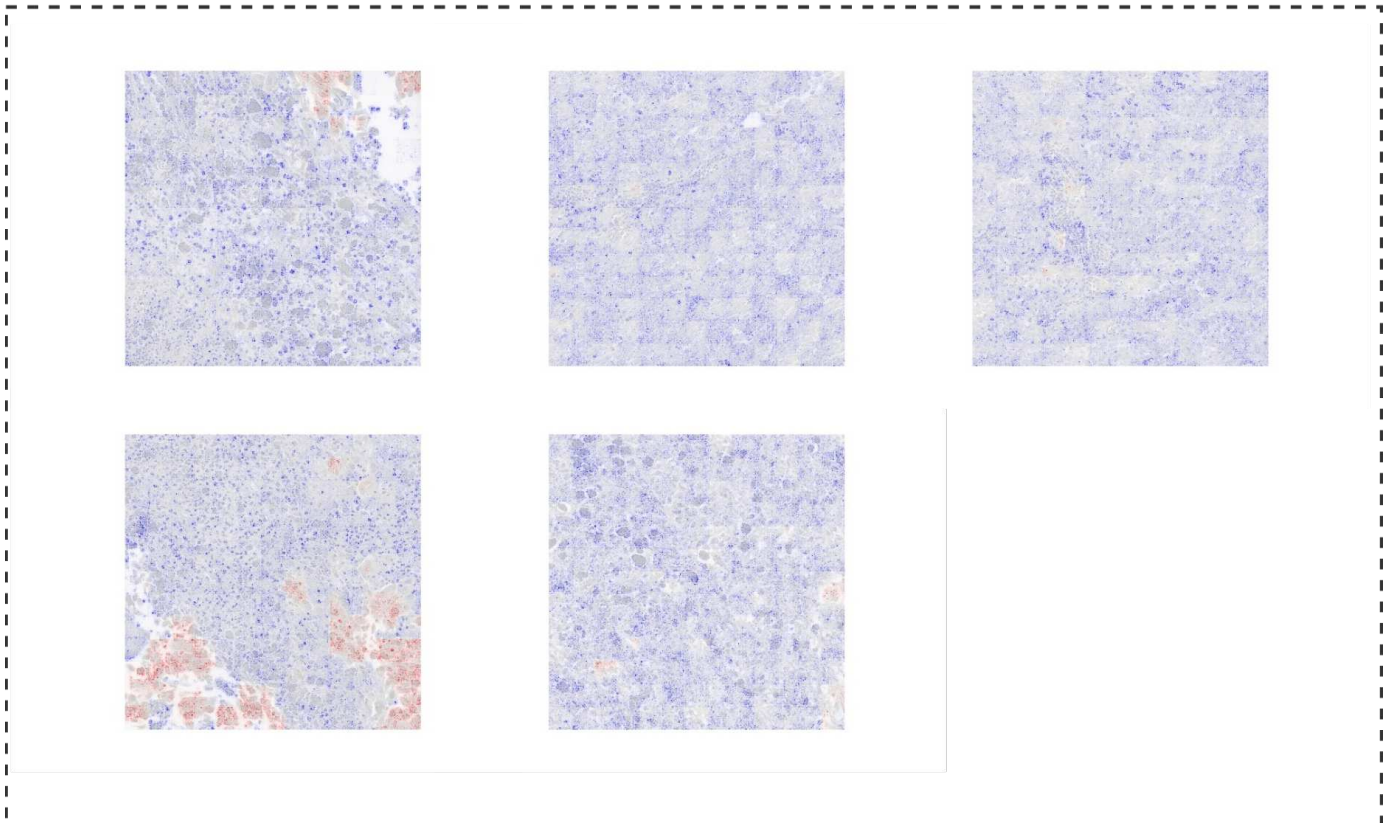


- test error unable to detect bad performance on missing subclasses

Hägele, Seegerer, Lapuschkin, Bockmayr, Samek, Klauschen, Müller, Binder, Resolving challenges in deep learning-based analyses of histopathological images using explanation methods,

Nat Sci Rep 2020

Training **with** necrosis samples.



your version1 labels and test set error cannot discover it

Hägele, Seegerer, Lapuschkin, Bockmayr, Samek, Klauschen, Müller, Binder, Resolving challenges in deep learning-based analyses of histopathological images using explanation methods,

Nat Sci Rep 2020

stop memorizing here again :)

Case B: improving model performance in small-sample size tasks:
Explanation-Guided Training for Cross-Domain Few-Shot Classification[2]
https://arxiv.org/abs/2007.08790

[2] J Sun, S Lapuschkin, W Samek, Y Zhao, NM Cheung, A Binder, ICPR 2020

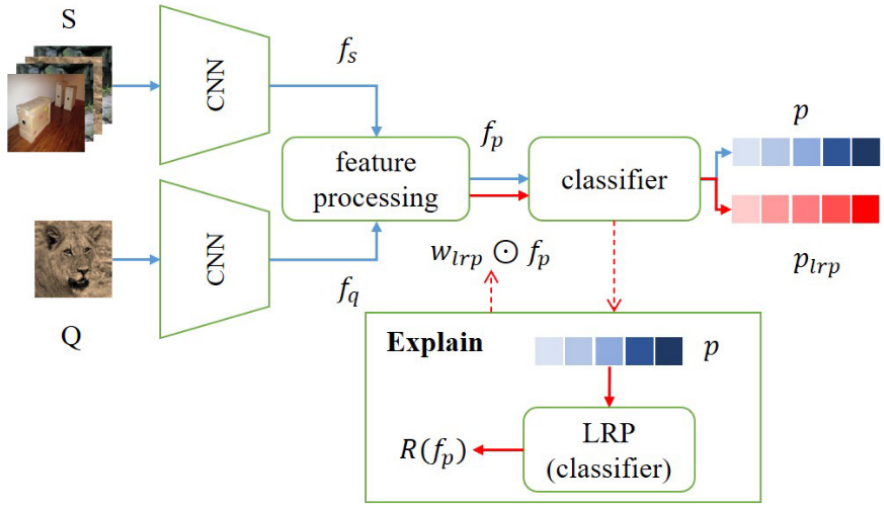
Motivation:

- Improve overall model performance beyond smaller modifications.
- Choose a low sample size setup with a somewhat challenging task.

Steps:

- compute prediction with original model $p(f)$ based on feature maps f
- compute explanation scores $R(\cdot)$ for selected feature maps $f \longmapsto \boldsymbol{R}(f) \in [-1, +1]^d$
- re-weight selected feature maps:

$$f_{lrp} = (1 + \boldsymbol{R}(f)) \odot f \quad (17)$$



- train: optimize sum of two losses: original features and reweighted features

$$L = L(y, p(f)) + \lambda L(y, p(f_{lrp})) \quad (18)$$

- prediction time: use unweighted features $p(f)$

- observation: consistent improvement (3 models, several datasets).

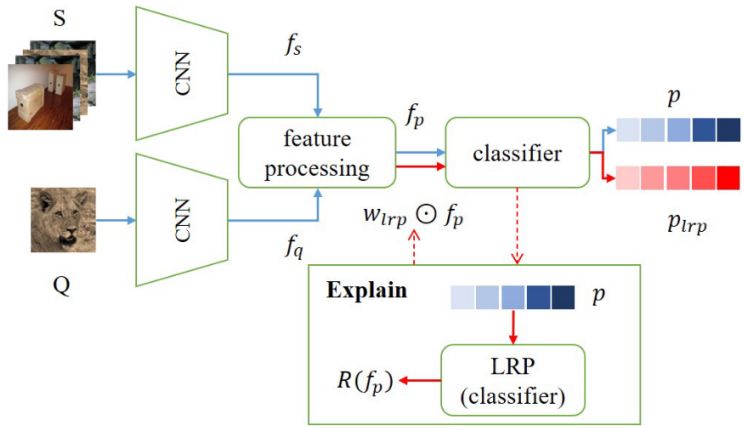
LRP-: explanation-guided training using LRP. *T*: transductive inference.

miniImagenet	1-shot	1-shot-T	5-shot	5-shot-T
RN	58.31±0.47%	61.52±0.58%	72.72±0.37%	73.64±0.40%
LRP-RN	**60.06±0.47%**	**62.65±0.56%**	**73.63±0.37%**	**74.67±0.39%**
CAN	**64.66±0.48%**	67.74±0.54%	79.61±0.33%	80.34±0.35%
LRP-CAN	64.65±0.46%	**69.10±0.53%**	**80.89±0.32%**	**82.56±0.33%**
mini-CUB	1-shot	1-shot-T	5-shot	5-shot-T
RN	41.98±0.41%	42.52±0.48%	58.75±0.36%	59.10±0.42%
LRP-RN	**42.44±0.41%**	**42.88±0.48%**	**59.30±0.40%**	**59.22±0.42%**
CAN	44.91±0.41%	46.63±0.50%	63.09±0.39%	62.09±0.43%
LRP-CAN	**46.23±0.42%**	**48.35±0.52%**	**66.58±0.39%**	**66.57±0.43%**
mini-Cars	1-shot	1-shot-T	5-shot	5-shot-T
RN	29.32±0.34%	28.56±0.37%	38.91±0.38%	37.45±0.40%
LRP-RN	**29.65±0.33%**	**29.61±0.37%**	**39.19±0.38%**	**38.31±0.39%**
CAN	31.44±0.35%	30.06±0.42%	41.46±0.37%	40.17±0.40%
LRP-CAN	**32.66±0.46%**	**32.35±0.42%**	**43.86±0.38%**	**42.57±0.42%**
mini-Places	1-shot	1-shot-T	5-shot	5-shot-T
RN	**50.87±0.48%**	**53.63±0.58%**	66.47±0.41%	67.43±0.43%
LRP-RN	50.59±0.46%	53.07±0.57%	**66.90±0.40%**	**68.25±0.43%**
CAN	56.90±0.49%	60.70±0.58%	72.94±0.38%	74.44±0.41%
LRP-CAN	**56.96±0.48%**	**61.60±0.58%**	**74.91±0.37%**	**76.90±0.39%**
mini-Plantae	1-shot	1-shot-T	5-shot	5-shot-T
RN	33.53±0.36%	33.69±0.42%	47.40±0.36%	46.51±0.40%
LRP-RN	**34.80±0.37%**	**34.54±0.42%**	**48.09±0.35%**	**47.67±0.39%**
CAN	36.57±0.37%	36.69±0.42%	50.45±0.36%	48.67±0.40%
LRP-CAN	**38.23±0.45%**	**38.48±0.43%**	**53.25±0.36%**	**51.63±0.41%**

- observation: consistent improvement (3 models, several datasets)

5-way 1-shot	miniImagenet	Cars	Places	CUB	Plantae
GNN	64.47±0.55%	30.97±0.37%	54.64±0.56%	46.76±0.50%	37.39±0.43%
LRP-GNN	**65.03±0.54%**	**32.78±0.39%**	**54.83±0.56%**	**48.29±0.51%**	**37.49±0.43%**
5-way 5-shot	miniImagenet	Cars	Places	CUB	Plantae
GNN	80.74±0.41%	42.59±0.42%	72.14±0.45%	63.91±0.47%	**54.52±0.44%**
LRP-GNN	**82.03±0.40%**	**46.20±0.46%**	**74.45±0.47%**	**64.44±0.48%**	54.46±0.46%

- combined with the feature transform from: (Cross-domain few-shot classification via learned feature-wise transformation, HY Tseng, HY Lee, JB Huang, MH Yang, ICLR 2020), it improves synergistically:



5-way 1-shot	Cars	Places	CUB	Plantae
RN	29.40±0.33%	48.05±0.46%	44.33±0.43%	34.57±0.38%
FT-RN	30.09±0.36%	48.12±0.45%	44.87±0.44%	35.53±0.39%
LRP-RN	30.00±0.32%	48.74±0.45%	45.64±0.42%	36.04±0.38%
LFT-RN	30.27±0.34%	48.07±0.46%	47.35±0.44%	35.54±0.38%
LFT-LRP-RN	**30.68±0.34%**	**50.19±0.47%**	**47.78±0.43%**	**36.58±0.40%**
5-way 5-shot	Cars	Places	CUB	Plantae
RN	40.01±0.37%	64.56±0.40%	62.50±0.39%	47.58±0.37%
FT-RN	40.52±0.40%	64.92±0.40%	61.87±0.39%	48.54±0.38%
LRP-RN	41.05±0.37%	66.08±0.40%	62.71±0.39%	48.78±0.37%
LFT-RN	41.51±0.39%	65.35±0.40%	64.11±0.39%	49.29±0.38%
LFT-LRP-RN	**42.38±0.40%**	**66.23±0.40%**	**64.62±0.39%**	**50.50±0.39%**

RelationNet. *FT* and *LFT* indicate the feature-wise transformation layer with fixed or trainable parameters.



There is no one optimal explanation. LRP works in practice, though.

Credit: Hanabusa Itcho