Aspects of explainable AI

Alexander Binder

University of Oslo (UiO)

April 24, 2021

UiO : Department of Informatics
University of Oslo

Learning goals

- 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
  - explanations of a single decision: see slide 8 for methods shown in this lecture
  - you do not need to memorize LRP formulas for the exam, but simple formulas like gradient times input

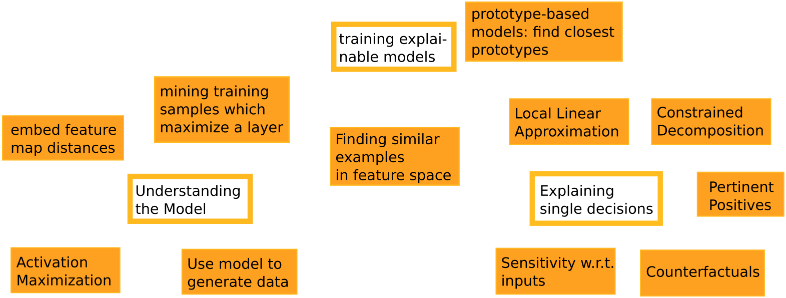
Outline

1. A coarse overview
2. t-SNE: visualize similarity of samples from a learned model by a 2D embedding
3. samples maximizing activations in a feature layer
4. Learning something from the metric defined by a feature map around a chosen sample: Finding similar examples for a given sample
5. per-sample explanations

...

⊙ ...

A taxonomy covering small parts



training explainable models

prototype-based models: find closest prototypes

mining training samples which maximize a layer

embed feature map distances

Finding similar examples in feature space

Understanding the Model

Activation Maximization

Use model to generate data

Local Linear Approximation

Constrained Decomposition

Explaining single decisions

Pertinent Positives

Sensitivity w.r.t. inputs

Counterfactuals

a broader view on different ways to explain

There are many ways to explain a prediction

- provide similar examples relative to a single sample (see literature on: similarity search, information retrieval)
- point to the most important regions for one single sample (compute scores for dimensions - single sample explanations)
- visualize relations between sets of samples - similarity embeddings (tSNE etc.)
- be able to answer questions of some fixed type (visual question answering? VQA with inputs and heatmaps?)
- be able to answer a wider range of questions using exploration of the environment (embodied QA?) or external sources (reasoning?)

Some of these settings like reasoning go a lot beyond direct explanations of predictions.

a broader view on different ways to explain

There are many ways to explain a prediction ... continuing:

- plug in the model to generate samples.
- provide examples which are similar but with a differing prediction (see literature on: counterfactuals https://arxiv.org/abs/1904.07451, pertinent negatives https://arxiv.org/abs/1802.07623)
- provide examples which are similar and contain the minimal content to have a similar prediction (pertinent positives https://arxiv.org/abs/1906.00117)
- training explainable models (whatever explainability means)

Some of these settings like reasoning go a lot beyond direct explanations of predictions.

a broader view on different ways to explain

Related to explainable AI, and very important for the success of products !!!

- bridging the gap to human perception, understanding, and usage:
  - visualization
  - human-computer interface design courses

ML/AI is not much worth without HCI.

In this lecture we can focus on a few aspects.

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)
  - Gradient, sensitivity, gradient times input
  - LIME
  - guided backprop
  - LRP and Deep Taylor
  - measuring the quality of your explanations
  - applications to finding biases in your data, to identify systematic failcases

Outline

1. A coarse overview
2. t-SNE: visualize similarity of samples from a learned model by a 2D embedding
3. samples maximizing activations in a feature layer
4. Learning something from the metric defined by a feature map around a chosen sample: Finding similar examples for a given sample
5. per-sample explanations

t-SNE for visualizing similarities between features

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?

t-SNE for visualizing similarities between features

One way is to visualize the similarities between samples by projecting each $x_i \in \mathbb{R}^d$ 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 $D_n = (x_1, \ldots, x_n)$, goal is to map each $x_i \in D_n$ onto a 2-dimensional data point y_i such that y_i, y_j which have similar distances to each other as the samples x_i, x_j from the set D_n.

t-SNE for visualizing similarities between features

- given two samples i, j with features x_i, x_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

$$p_{j|i} \propto \exp(-\|x_i - x_j\|^2 / 2\sigma^2)$$

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

- 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 $p_{i|j} \approx 0$.

t-SNE for visualizing similarities between features

What did we obtain?

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

next step:

- find for x_i a synthetic sample $y_i \in \mathbb{R}^2$ and a model of interaction strength q_{ij} between low-dimensional representatives y_i and y_j

t-SNE for visualizing similarities between features

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

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

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

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

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

Why for the y_i use a heavy tailed probability?

t-SNE for visualizing similarities between features

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}.

t-SNE for visualizing similarities between features

What means choosing a heavy tailed probability for the model of the y_i? A heavy tailed probability assigns a relatively high probability to points far away. Therefore those points x_i, x_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 p_{ij}.)

By this t-sne can focus on putting only those points i and j close in the embedding y_i, y_j 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.

Outline

1. A coarse overview
2. t-SNE: visualize similarity of samples from a learned model by a 2D embedding
3. samples maximizing activations in a feature layer
4. Learning something from the metric defined by a feature map around a chosen sample: Finding similar examples for a given sample
5. per-sample explanations

Analyzing learned features by dataset evaluation

Consider those images/ input samples $\hat{x}$ which activate a channel c of a feature map z most

$$s_c^{(1)}(x) = \sum_{h,w} z^2[0,c,h,w](x)$$

$$s_c^{(2)}(x) = \sum_{h,w} \max(0, z[0,c,h,w])(x)$$

$$\hat{x} = \mathrm{argmax}_{x \in \mathsf{Data}} s_c(x)$$

It shows what a channel focuses most on in your given dataset. If the dataset has the same distribution, as your training dataset, then one can claim to infer what the feature layer has learned from the training data (why this is not valid if using another dataset?)

- order inputs x according to $s(x)$ in descending order and visualize them
- explains properties of feature map, not a single decision or relations in a set of samples.

Outline

1. A coarse overview
2. t-SNE: visualize similarity of samples from a learned model by a 2D embedding
3. samples maximizing activations in a feature layer
4. Learning something from the metric defined by a feature map around a chosen sample: Finding similar examples for a given sample
5. per-sample explanations

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. $h(\cdot) = h_w(\cdot)$
- 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 and dataset driven activation maximization: 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

Outline

1. A coarse overview

2. t-SNE: visualize similarity of samples from a learned model by a 2D embedding

3. samples maximizing activations in a feature layer

4. Learning something from the metric defined by a feature map around a chosen sample: Finding similar examples for a given sample

5. per-sample explanations

Gradient

coarse idea:

- 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 $f(x)$?

Gradient

- ⊙ compute gradient
- ⊙ use its square of a partial derivative $\frac{\partial f}{\partial x_d}(x)$ as a score for the relevance of an input dimension x_d:

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

- (+) easy to implement
- (+) fast to compute
- (−) see below what sensitivity explains, often not the question you wanted to ask

Gradient

The main drawback:

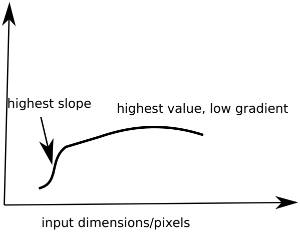
What the gradient explains

- 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 $\neq$ most contributing

Gradient

Compare: where a function has highest value vs where a function has highest slope.



A math example:

$$f(x) = w \cdot x$$
$$\frac{\partial f}{x_d}(x) = w_d$$

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

Gradient × Input

Use as explanation:

$$r_d(x) = \frac{\partial f}{x_d}(x)\, x_d$$
$$\mathbf{R} = \nabla f(x) \cdot \mathbf{x}$$

(+) works well for shallow smaller nets (LeNet) and sigmoid networks

(+) motivation as a heuristic: close to Taylor decomposition as explanation for a point x_0 close to the point x to be explained, if x_0 is chosen orthogonal to the gradient ($\nabla f(x) \cdot x_0 = 0$).

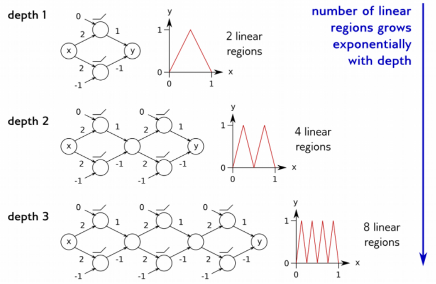
$$f(x_0) \approx f(x) + \nabla f(x)(x_0 - x) + \mathcal{O}(\|x - x_0\|^2)$$
$$f(x) \approx f(x_0) + \nabla f(x)(x - x_0) + \mathcal{O}(\|x - x_0\|^2)$$

This is an explanation relative to a point x_0 such that $\|x - x_0\| < \epsilon$ Further ignore the constant contribution $f(x_0)$ in the Taylor series because it is independent of input dimension for the explanation, to arrive at “Gradient × Input”.

Gradient shattering

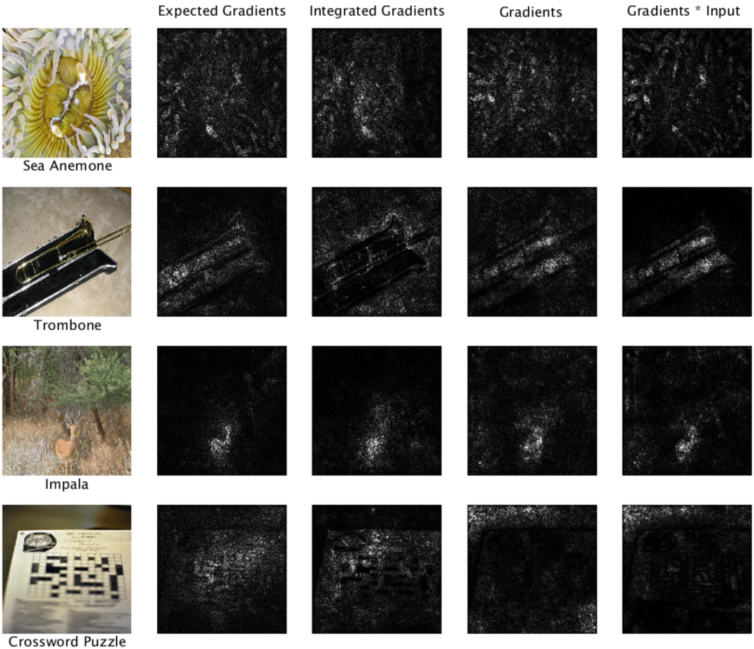
(–) can be uninformative for deep ReLU-networks due to gradient shattering problem:
Montufar NIPS 2014 https://papers.nips.cc/paper/5422-on-the-number-of-linear-regions-of-deep-neural-networks.pdf.
Balduzzi 2017
http://proceedings.mlr.press/v70/balduzzi17b/balduzzi17b.pdf. See:

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



Worth reading: works of Marco Ancona

Gradient shattering



Interpret Single Decisions III: Integrated gradient

Integrated Gradient: *Axiomatic attribution for deep networks*
Sundararajan et al., *ICML 2017*

a heuristic very similar to the gradient times input idea:

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

- ⊙ 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 × input? Averaging gradients smoothes out the gradient shattering.
- (+)(−) IG gets better and slower when hundreds of points are used (+ slows down).

Does gradient shattering really affect explanations?

Understanding Integrated Gradients with SmoothTaylor for Deep Neural Network Attribution (not introducing a new explanation method)[1]

- Used (global) Taylor approximation with averaged multiple roots with fixed variance σ^2 around x

$$R_d(x) = \frac{1}{R}\sum_{r=1}^{R}(x_d - z_d^{(r)})\,\frac{\partial f}{\partial x_d}\bigg|_{z^{(r)}},\ z^{(r)} \sim N(x, \sigma^2 I) \qquad (1)$$

- measure explanation quality by correlation to predictor scores under averaged iterative region alteration
- measure smoothness by total variation

[1] GSW Goh, S Lapuschkin, L Weber, W Samek, A Binder, ICPR 2020

Does gradient shattering really affect explanations?

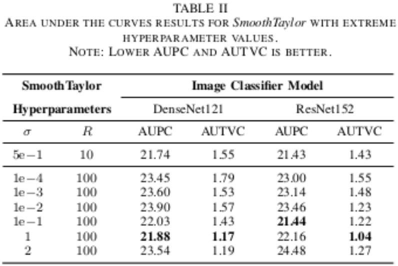


TABLE II
AREA UNDER THE CURVES RESULTS FOR *SmoothTaylor* WITH EXTREME HYPERPARAMETER VALUES.
NOTE: LOWER AUPC AND AUTVC IS BETTER.

SmoothTaylor		**Image Classifier Model**			
Hyperparameters		DenseNet121		ResNet152	
σ	R	AUPC	AUTVC	AUPC	AUTVC
5e−1	10	21.74	1.55	21.43	1.43
1e−4	100	23.45	1.79	23.00	1.55
1e−3	100	23.60	1.53	23.14	1.48
1e−2	100	23.90	1.57	23.46	1.23
1e−1	100	22.03	1.43	**21.44**	1.22
1	100	**21.88**	**1.17**	22.16	**1.04**
2	100	23.54	1.19	24.48	1.27

- Explanation quality and total variation smoothness degenerates when the sampling variance is very small. Contradicts the idea of being close to x for good Taylor approximations.
- Therefore: To average gradients at points sufficiently far away is important. Evidence for the effect of gradient shattering. Explains why IG works better with many roots than gradient times input.

Interpret Single Decisions IV: Lime

Ribeiro et al, ICML 2016 https://arxiv.org/pdf/1602.04938.pdf

The first thing to understand: the decisions made by a linear model are easily explainable

$$g(x) = \sum_d w_d x_d$$

$$r_d := w_d x_d$$

The contribution of a single decision can be explained easily. If it has a bias, there is an unexplained component though – the bias b, which cannot be naturally assigned into contributions of single dimensions d:

$$g(x) = \sum_d w_d x_d + b$$

$$r_d := w_d x_d$$

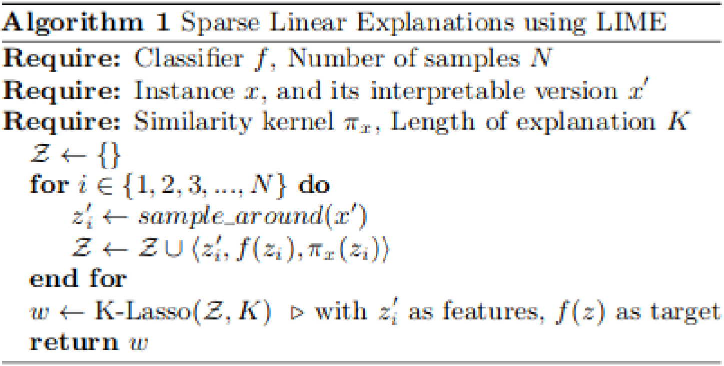
Interpret Single Decisions IV: Lime

The idea of Lime: given a test sample x learn a locally linear approximation to f around x.

$$f(x) \approx A(x) = \sum_d w_d x_d$$

$$r_d(x) = w_d x_d$$

How to get to the locally linear approximation $A(x)$?



Algorithm 1 Sparse Linear Explanations using LIME

```
Require: Classifier $f$, Number of samples $N$
Require: Instance $x$, and its interpretable version $x'$
Require: Similarity kernel $\pi_x$, Length of explanation $K$
  $\mathcal{Z} \leftarrow \{\}$
  for $i \in \{1, 2, 3, ..., N\}$ do
      $z'_i \leftarrow sample\_around(x')$
      $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i) \rangle$
  end for
  $w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$  ▷ with $z'_i$ as features, $f(z)$ as target
  return $w$
```

Interpret Single Decisions IV: Lime

K-Lasso

- train lasso

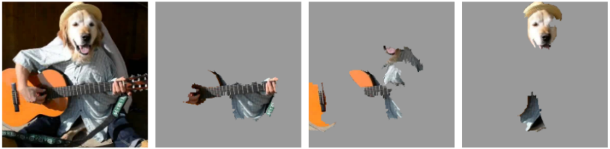
$$w \leftarrow \mathrm{argmin}_w \frac{1}{2n} \sum_i (f(z_i) - w \cdot z_i)^2 + \lambda \|w\|_1$$

- ℓ_1-norm induces sparsity (makes many weights to be zero)
- select K dimensions with highest weights
- train a linear/ridge regression with only those dimensions to obtain $A(x)$
- parameters: sampling radius size, K, λ

A large sampling radius allows to learn correlations between neighboring data points more than just the gradient.
One thing to be taken care is: for a too small sampling radius LIME converges to the gradient – which answers a different question.

Interpret Single Decisions IV: Lime

(+) good for problems where using a sparse subset of features makes well sense, e.g. tabular/ columnar data (finance, lab measurements). Some setups do not go well with sparse subsets:



(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

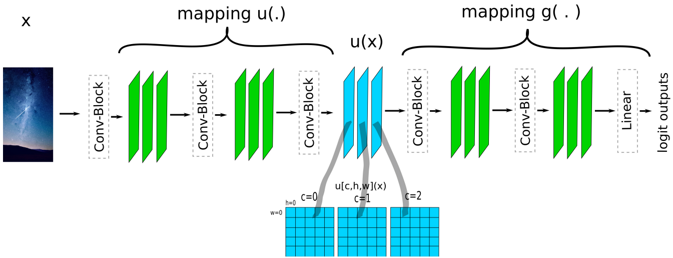
(−) need to train a model for every input sample

(−) explanation sensitive to radius parameter – need to validate / test choices of this parameter

Interpret Single Decisions V: Grad-CAM

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

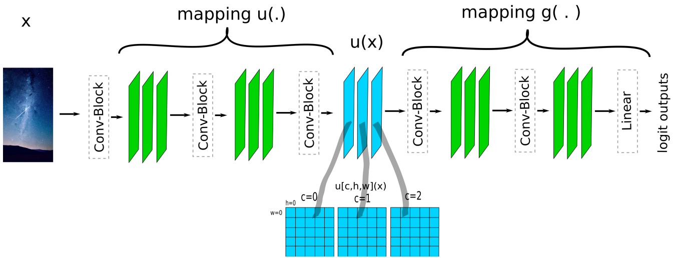
- given a feature map $u(x)$ with components $u[c,h,w](x)$ having C channels, width W, height H, and a predictor $f(x) = g(u(x))$ where $g(\cdot)$ are some layers on top of u.



Gradient times input in the space of u would be:

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

Interpret Single Decisions V: Grad-CAM



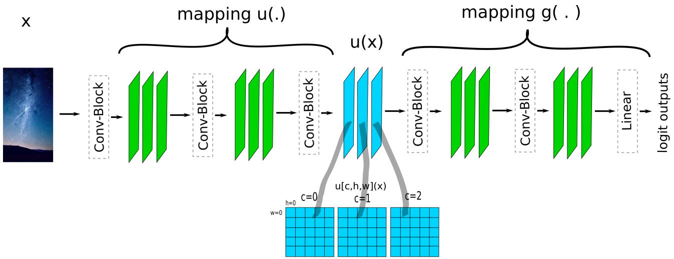
Gradient times input in the space of u would be:

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

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

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

Interpret Single Decisions V: Grad-CAM



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 $ReLU$

Interpret Single Decisions V: Grad-CAM

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

Interpret Single Decisions V: Grad-CAM

- ⊙ 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?)

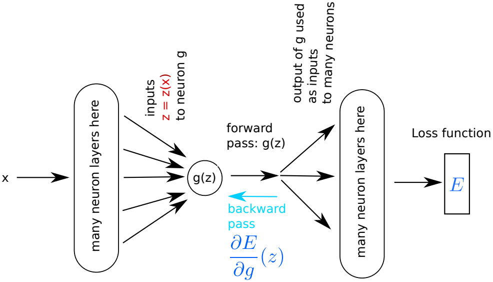
(+) simple to implement

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

Interpret Single Decisions VI: Guided backpropagation

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.



Interpret Single Decisions VI: Guided backpropagation

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

- the inputs 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?

- if the activation of the neuron is negative $g(z) < 0$, then $g(z)$ is a suppressing neuron. rule: Ignore gradients from suppressing neurons, pass through only gradient signal from firing 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

Interpret Single Decisions VI: Guided backpropagation

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

Interpret Single Decisions VII: Occlusion

- 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

Interpret Single Decisions VII: Occlusion

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?

Interpret Single Decisions VII: Occlusion

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

Interpret Single Decisions VII: Occlusion

- (+) 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

Interpret Single Decisions

- Starting point of many methods: a prediction $f(x)$ over a sample $x = (x_1, \dots, x_d, \dots, x_D)$
- Many methods compute a score $r_d(x)$ such that $r_d(x)$ tells how much one dimension x_d from x contributes to the prediction
- one way to achieve this, is first order Taylor decomposition around some point $x^{(0)}$

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

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

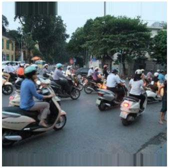
- linearizes relatively to $x^{(0)}$. Explains the difference of prediction in x relative to the prediction made in $x^{(0)}$. Note: a relative explanation!
- the next method applies such an idea for every layer, and stacks decompositions

Interpret Single Decisions VIII: LRP

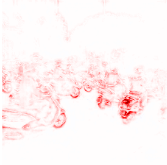
- given: A. trained model f, B. a prediction $f(x)$ for input $x = (x_1, \ldots, x_d, \ldots, x_D)$.
- general case: To compute a relevance score $r_d(x)$ for every input dimension x_d of input x explaining the prediction $f(x)$, such that approximately:

$$f(x) \approx \sum_{d=1}^{D} r_d(x) \leftarrow \text{decomposition with constraints} \tag{2}$$

image

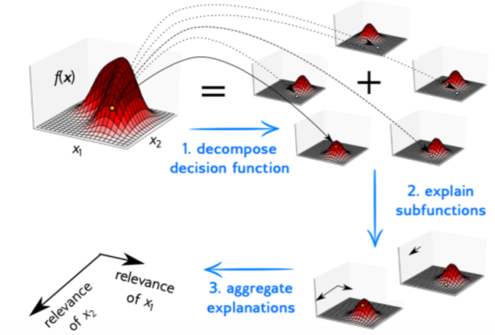


LRP-α-β



LRP: Deep Taylor Decomposition

- Divide and conquer: decompose network in layers

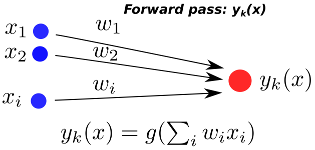


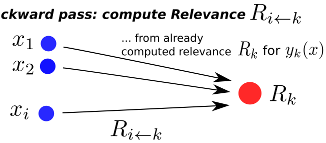
- Taylor approximation per layer/neuron
- easier to find roots for one layer
- better robustness to gradient shattering for certain choices of $x^{(0)}$

Relevance computation for one neuron

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

Relevance distribution for one neuron: example β-rule




Backward pass: compute Relevance $R_{i\leftarrow k}$
x_1
x_2
x_i
... from already computed relevance R_k for $y_k(x)$
R_k
$R_{i\leftarrow k}$

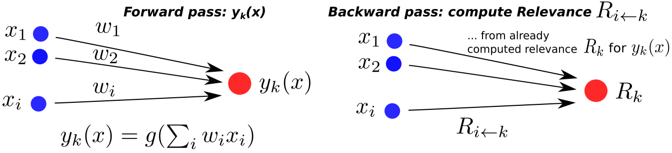
LRP-β: given: have computed already R_k as relevance of neuron output $z_k = \sum_i w_{ik} x_i + b$,

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

$$R_{i\leftarrow k}(\mathbf{x}) = R_k \left((1+\beta) \frac{(w_{ik}x_i)_+}{\sum_{i'} (w_{i'k}x_{i'})_+} - \beta \frac{(w_{ik}x_i)_-}{\sum_{i'} (w_{i'k}x_{i'})_-} \right) \qquad (3)$$

- β controls ratio of negative to positive evidence $\frac{\beta}{1+2\beta}$.
- negative to total evidence: $\frac{\beta}{1+2\beta} \overset{\beta\to\infty}{\to} 0.5$,
  It is fixed independent of network inputs(!).
- bounded relevance scale: $|R_{i\leftarrow k}| \leq (1+\beta)|R_k|$ (not true for LRP-ϵ!)

Relevance computation for one neuron

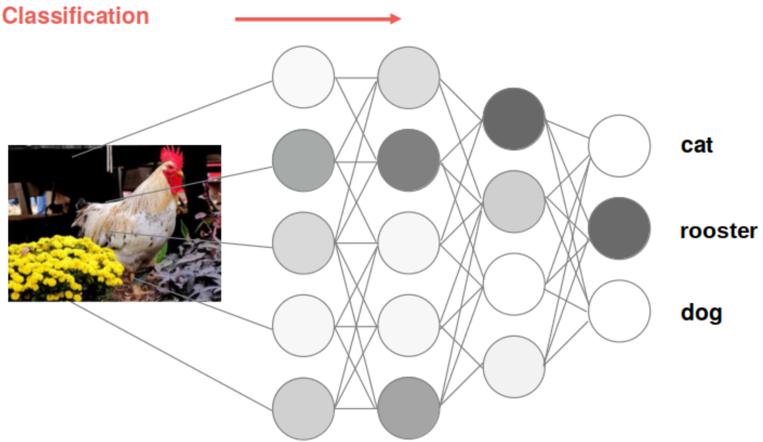

Forward pass: $y_k(x)$
x_1
x_2
x_i
w_1
w_2
w_i
$y_k(x)$
$y_k(x) = g(\sum_i w_i x_i)$
Backward pass: compute Relevance $R_{i\leftarrow k}$
x_1
x_2
x_i
... from already computed relevance R_k for $y_k(x)$
R_k
$R_{i\leftarrow k}$

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

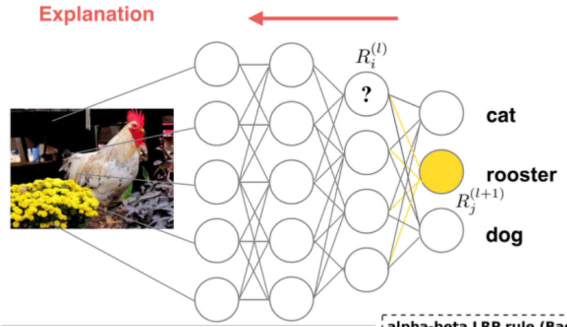
$$R_i := \sum_{k:i \text{ is input to } k} R_{i\leftarrow k} \tag{4}$$

Same as in backpropagation

What could be a possible explanation of a prediction?



What could be a possible explanation of a prediction?



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

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

where $\alpha + \beta = 1$

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

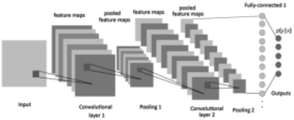
which root for which layer?

Name	Formula	layers
LRP-ϵ	$\sum_k R_k \left(\frac{x_i w_{ik}}{\sum_i x_i w_{ik}+b+\epsilon\,\mathrm{sign}(z)}\right)$	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

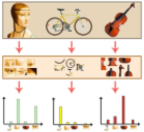
Its a mere serving suggestion. Define a loss and measure the quality of your explanations and choose by that!

LRP Applied to Variety of Models

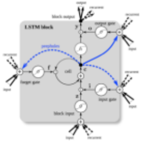
Convolutional NNs
(Bach'15, Binder'16, Arras'17 ...)



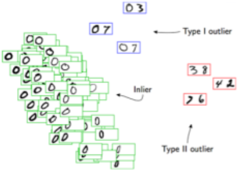
BoW models
(Bach'15, Lapuschkin'17 ...)



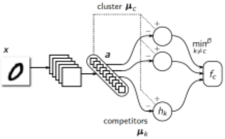
LSTM
(Arras'17, Arras ...
Hochreiter et al. 2019)



One-class SVM
(Kauffmann'18)



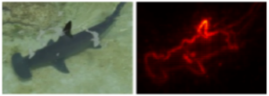
Clustering
(Kauffmann'19)



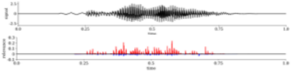
explaining unsupervised learning

LRP Applied to Variety of Tasks

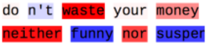
General Images (Bach’ 15, Lapuschkin’16)



Speech (Becker’18)



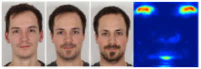
Text Analysis (Arras’16 &17)



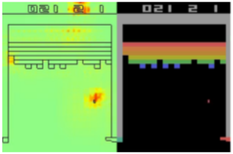
do n't waste your money

neither funny nor susper

Morphing (Seibold’18)



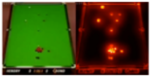
Games (Lapuschkin’19)



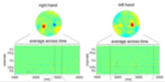
VQA (Samek’19)

there is a metallic cube ; are there any large cyan metallic objects behind it ?

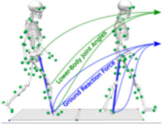
Video (Anders’18)



EEG (Sturm’16)



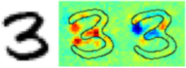
Gait Patterns (Horst’19)



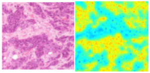
Faces (Lapuschkin’17)

Digits (Bach’ 15)

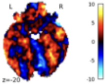
Image Class '3' Class '9'



Histopathology (Hägele’19)



fMRI (Thomas’18)



Interpret Single Decisions VIII: LRP

No method is better than all others on all reasonable use cases, which includes LRP/Deep Taylor

(−) 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

(+) good results for several applications / evaluation papers with certain presets (ignore biases, use hybrid LRP-β, γ), RNNs such as LSTMs, high-dimensional data

(+) reasonably fast 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)*

(+) tunable resolution heatmaps, lots of tuning options

Interpret Single Decisions VIII: LRP

LRP experience:

- linear layers: LRP-ϵ
- conv-layers: LRP-β,LRP-γ
- attention weighted layers:

$$f = \sum_i w_i(v) v_i$$

treat weights $w_i(v)$ as if they were constant weights, apply LRP-ϵ to the resulting linear layer in v_i obtain relevances $R(v_i)$ for v_i

$$f = \sum_i w_i v_i$$

$$R(f) \longmapsto R(v_i)$$

Which method to choose ?

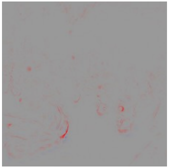
No method is better than all others on all reasonable use cases.

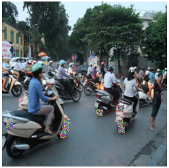
- if the performance measure is sensitivity under small changes of a single dimension/pixel, then you need only the gradient!
- can we evaluate somehow ?

how to evaluate the quality?

One approach: 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)

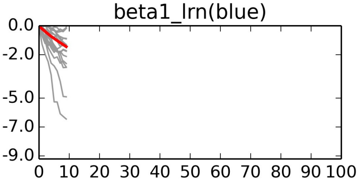
- choose region size, compute average heatmap score for each region
- Idea: use relevance scores (from any method) r to order regions.
- *modify* regions in the implied order
- measure decrease of prediction $f(\mathbf{x})$ under modifications
- key idea: if most important pixels get the highest score,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





Evaluation?

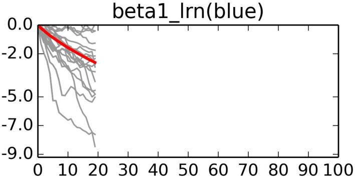
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

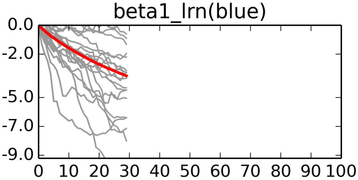
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

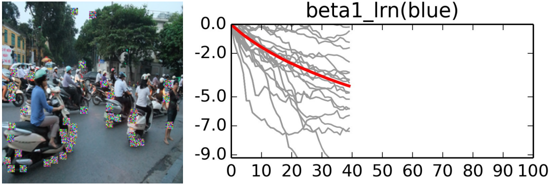
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

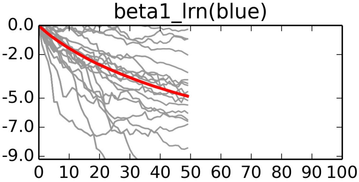
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

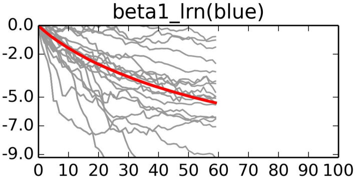
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

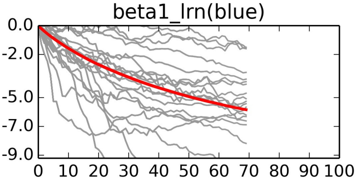
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

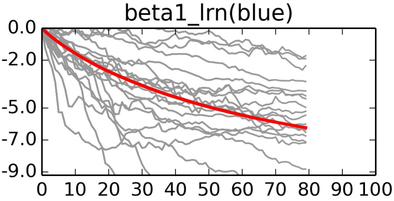
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

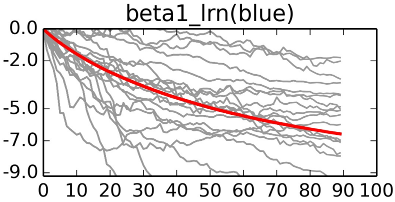
- Idea: measure area under graph - averaged over many runs and images (red line)



- can compare against random orderings of pixels

Evaluation?

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

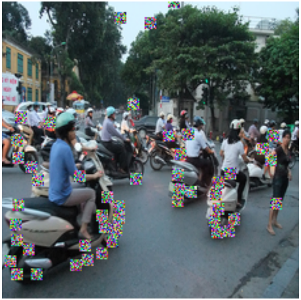


- can compare against random orderings of pixels

Evaluation?

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

how to evaluate the quality?

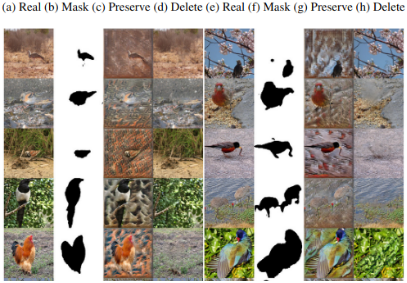
https://ieeexplore.ieee.org/abstract/document/7552539, https://arxiv.org/abs/1509.06321 (disclaimer: own work)

- 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 the like (see above).

Challenges in explainability for vision

- 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



how to evaluate the quality?

- correlation to human intuition
- correlation to bounding boxes
- 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)

The value of explanations (not just LRP...)

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

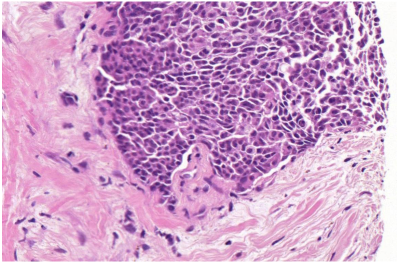
B. improving model performance in small-sample size tasks: LRP-guided training to improve cross-domain few shot learning

The value of explanations (not just LRP...)

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

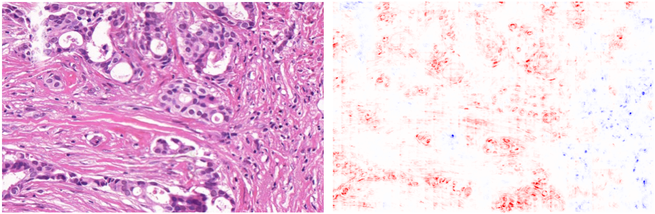
Iterative Dataset Design (Medical Data)

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



Iterative Dataset Design (Medical Data)

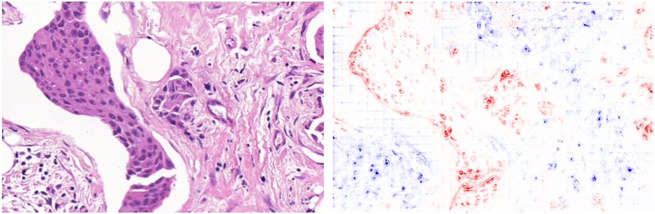
- improve model via growing the dataset
- decide what unlabeled data to add into next iteration of train and test set



- Interpretability for efficiency in the selection step before labelling!

Iterative Dataset Design (Medical Data)

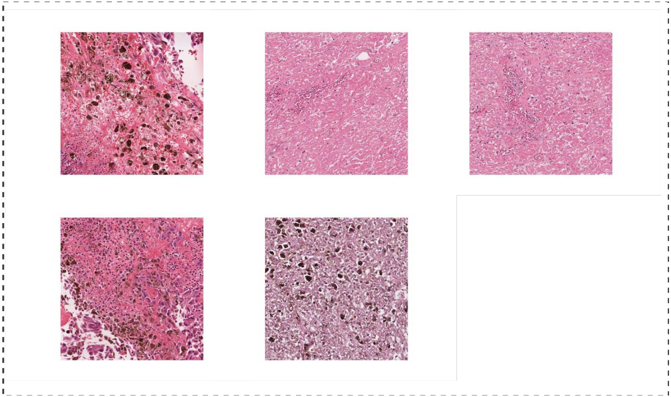
- improve model via growing the dataset
- decide what unlabeled data to add into next iteration of train and test set – precursor to labelling.



- Interpretability for efficiency in the selection step before labelling!

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

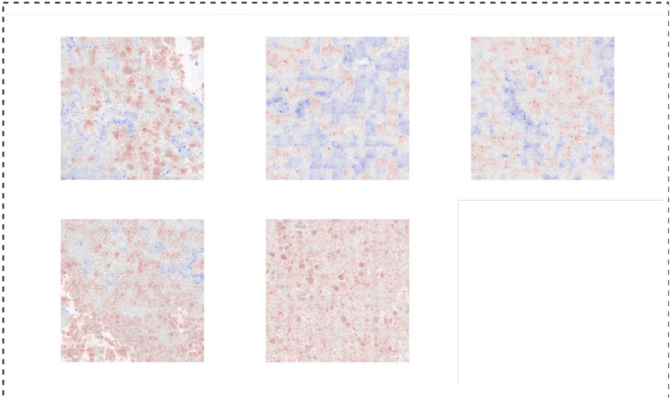
Original HE images with necrosis regions:



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.

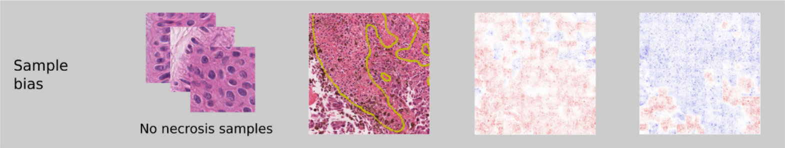


Hägele, Seegerer, Lapuschkin, Bockmayr, Samek, Klauschen, Müller, Binder,
Resolving challenges in deep learning-based analyses of histopathological images using explanation methods,
Nat Sci Rep 2020

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)

training image	test image	heatmap with bias	heatmap w/o bias

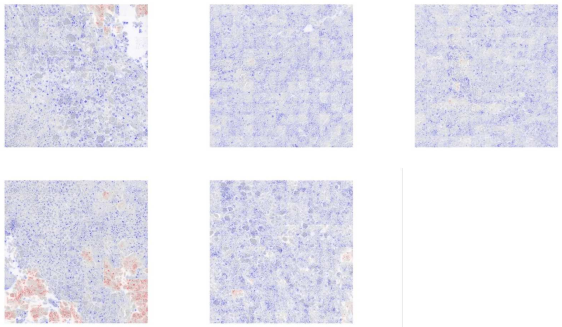


- test error unable to detect bad performance on missing subclasses

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

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

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

The value of explanations (not just LRP...)

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

Explanation-Guided Training for Cross-Domain Few-Shot Classification

Motivation:

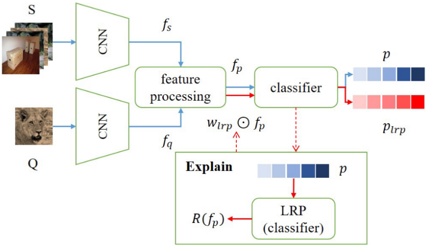
- Improve overall model performance beyond smaller modifications.
- Choose a low sample size setup with a somewhat challenging task.

Explanation-Guided Training for Cross-Domain Few-Shot Classification

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:

$$f_{lrp} = (1 + \boldsymbol{R}(f)) \odot f \qquad (5)$$



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

$$L = L(y, p(f)) + \lambda L(y, p(f_{lrp})) \qquad (6)$$

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

Explanation-Guided Training for Cross-Domain Few-Shot Classification

- 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%**

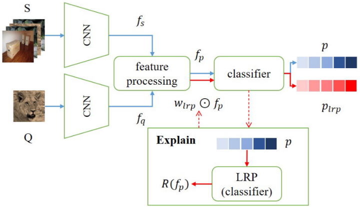
Explanation-Guided Training for Cross-Domain Few-Shot Classification

- 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%

Explanation-Guided Training for Cross-Domain Few-Shot Classification

- 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:

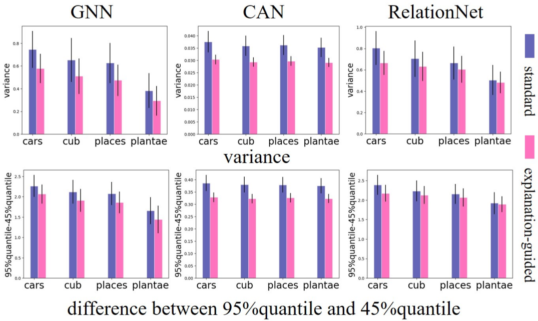


5-way 1-shot	Cars	Places	CUB	Plantae
RN	29.40±0.33%	48.05±0.46%	44.33±0.43%	34.57±0.38%
FT-RN	30.09±0.36%	48.12±0.45%	44.87±0.44%	35.53±0.39%
LRP-RN	30.00±0.32%	48.74±0.45%	45.64±0.42%	36.04±0.38%
LFT-RN	30.27±0.34%	48.07±0.46%	47.35±0.44%	35.54±0.38%
LFT-LRP-RN	**30.68±0.34%**	**50.19±0.47%**	**47.78±0.43%**	**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.

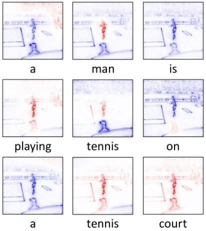
Explanation-Guided Training for Cross-Domain Few-Shot Classification



- LRP-weighted training makes the filter activation spectrum more uniform across channels (visible by lower quantile differences)

Teaser: Explaining Image captioning

- Words are generated often by recurrent neural networks: $\text{word}_{n+1} = f(\text{Image}, \text{word}_1, \text{word}_2, \ldots, \text{word}_n)$
- LRP can be applied to RNNs such as LSTM.

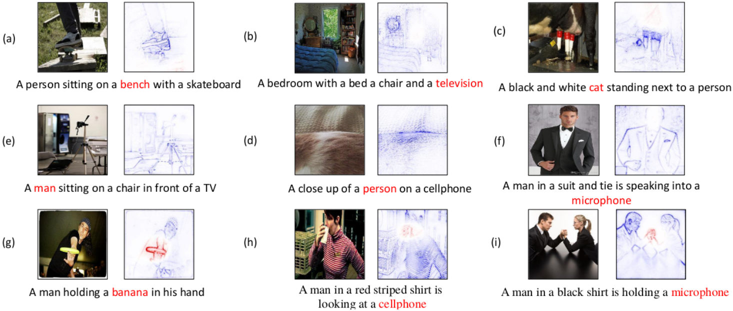


2.Man: a
3.Is: a man
4.Playing: a man is
5.Tennis: a man is playing
6.On: a man is playing tennis
7.A: a man is playing tennis on
8.Tennis: a man is playing tennis on a
9.Court: a man is playing tennis on a tennis

0.75,1	0.5,0.75	0.25,0.5	0,0.25
-0.25,0	-0.5,-0.25	-0.75,-0.5	-1,-0.75

https://arxiv.org/abs/2001.01037

Teaser: Explaining Image captioning



(a) A person sitting on a bench with a skateboard

(b) A bedroom with a bed a chair and a television

(c) A black and white cat standing next to a person

(e) A man sitting on a chair in front of a TV

(d) A close up of a person on a cellphone

(f) A man in a suit and tie is speaking into a microphone

(g) A man holding a banana in his hand

(h) A man in a red striped shirt is looking at a cellphone

(i) A man in a black shirt is holding a microphone

https://arxiv.org/abs/2001.01037

Questions?!