Explainable AI: Concepts, Methods and Applications

Wojciech Samek
TU Berlin & Fraunhofer HHI
Syllabus

Part I
- Why explainability?
- What to explain?
- Overview of XAI techniques
- Implementation & codes

Part II
- Theoretical Embedding
- Evaluating explanations
- Applications of XAI
- XAI beyond explaining

Part III
- Extensions beyond NN
- Concept-level explanations
- Outlook & discussion
"We might consider the perceptron as a black box, with a TV camera for input, and an alphabetic printer or a set of signal lights as output."
"We might consider the perceptron as a **black box**, with a TV camera for input, and an alphabetic printer or a set of signal lights as output."

Frank Rosenblatt (1928 - 1971)
NN are Black Boxes

Although artificial neural networks are employed in an ever growing variety of applications, their inner workings are still viewed as a black box, which is due to the complexity of the non-linear dynamics that govern neural network learning. The key parameters in this learning

**Abstract** — Artificial neural networks are efficient computing models which have shown their strengths in solving hard problems in artificial intelligence. They have also been shown to be universal approximators. Notwithstanding, one of the major criticisms is their being black boxes, since no satisfactory explanation of their behavior has been offered. In this paper,

Neural networks, which make no assumption about data distribution, have achieved improved image classification results compared to traditional methods. Unfortunately, a neural network is generally perceived as being a ‘black box’. It is extremely difficult to document how specific classification decisions are reached. Fuzzy systems, on the other hand,
NN are Black Boxes

In last decade exponential growth in XAI papers.
NN are Black Boxes

Can we trust ML models without understanding what they do?

prediction: "horse"
NN are Black Boxes

\[
\min_{f \in F} \int_{x,y} \| f(x) - y \|^2 \, d\rho(x, y)
\]

Is minimizing the error a guarantee for the model to work well in practice?
"Explainability is neither necessary nor sufficient for trust. [...] I only trust thorough testing."
"Explainability is neither necessary nor sufficient for trust. [...] I only trust thorough testing."

Yann LeCun (1960)
PASCAL VOC Challenge (2005 - 2012)

Task: Multi-label classification for 20 object classes.

The VOC2021 train/val data has 11,530 images and 31,561 objects.
PASCAL VOC Challenge (2005 - 2012)
Unmasking Clever Hans Predictors

Leading method (Fisher-Vector / SVM Model) of PASCAL VOC challenge
Unmasking Clever Hans Predictors

Leading method (Fisher-Vector / SVM Model) of PASCAL VOC challenge

(Lapuschkin et al. 2016 & 2019)
Unmasking Clever Hans Predictors

‘horse’ images in PASCAL VOC 2007
The Real Clever Hans

Clever Hans (German: der Kluge Hans; fl. 1907) was a horse that was claimed to have performed arithmetic and other intellectual tasks. After a formal investigation in 1907, psychologist Oskar Pfungst demonstrated that the horse was not actually performing these mental tasks, but was watching the reactions of his trainer.
Back Our Question

Is minimizing the error a guarantee for the model to work well in practice?

--> NO in the case of PASCAL VOC Challenge

Note: Please keep in mind that some of the best scientists were involved in preparing the challenge or participated in it. These flaws remained undetected for over one decade.

Are we better off with ImageNet today?
Thorough testing is very difficult in practice and may require large, representative (!) datasets. Explanations can help to foster trust by identifying misbehaviour from just a single example.
Why explainability?

We need explainability in order to:

- trust & verification
- legal aspects
- improve system
- learn from the system
Why explainability?

We need explainability in order to:

- trust & verification

"ML medical diagnosis system misclassifies patient’s disease …"
Why explainability?

We need explainability in order to:

- trust & verification
- legal aspects

“right to explanation”

- avoid discrimination

Retain human decision in order to assign responsibility.
Why explainability?

“We need explainability in order to:

To improve the system and learn from the system.

“'It's not a human move. I've never seen a human play this move.” (Fan Hui)
Why explainability?

Learn about the physical / biological / chemical mechanisms. (e.g. find genes linked to cancer)
Why explainability?

We need explainability in order to:

- Generalization error + human experience
- Improve system
- Learn from the system
- Improve system

Model/data improvement
Interpretability
Human inspection
Verified predictions

Generalization error + human experience
What to explain?
"to explain" means to make something clear or easy to understand by describing or giving information about it.

(Cambridge Dictionary)
"to explain" means to make something clear or easy to understand by describing or giving information about it.

(Cambridge Dictionary)

What and relative to what?

To whom?
Explain vs. Interpret vs. Understand

**explaining**: refers to the process of computation of the explanation (e.g. attribution map).

**interpreting**: refers to the process of assigning a meaning to the explanation.

**understanding**: refers to a deeper functional insight of model.
Explain vs. Interpret vs. Understand

prediction: "25-32 years old"

explain: "compute heatmap"

interpret: "laughing is important for prediction"

(Lapuschkin et al. 2017)
Explain vs. Interpret vs. Understand

- **Prediction:** "over 60 years old"
- **Explain:** "compute heatmap"
- **Interpret:** "laughing speaks against prediction"
Explain vs. Interpret vs. Understand

understand: "ML model treats laughing as feature for discriminating age"

interpret: "laughing speaks against prediction"

explain: "compute heatmap"

prediction: "over 60 years old"
Would you find this bias just with testing?

I doubt it.
What Do We Explain?

- **prediction**: “Explain why a certain pattern $x$ has been classified in a certain way $f(x)$.”
- **model**: “What concept does a particular neural encode?”
- **data**: “Which dimensions of the data are most relevant for the task.”
Explaining Classification Decisions

“why a given image is classified as a pool table”

post-hoc!
Explaining Other Tasks?

Segmentation

Regression

Outlier Detection

Generative models

Reinforcement Learning
Overview of XAI techniques
Brief Recent History of Post-Hoc XAI

Visualization of neural networks using saliency maps
NJS Morch, U Kjems, LK Hansen... - Proceedings of ICNN ..., 1995

How to explain individual classification decisions
D Baehrens, T Schroeter, S Harmeling... - The Journal of Machine ..., 2010 - jmlr.org

Deep inside convolutional networks: Visualising image classification models and saliency maps
Brief Recent History of Post-Hoc XAI

On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation
S. Bach, A. Binder, G. Montavon, F. Klauschen... - PloS one, 2015 - journals.plos.org

“Why should I trust you?” Explaining the predictions of any classifier
MT. Ribeiro, S. Singh, C. Guestrin - Proceedings of the 22nd ACM ..., 2016 - dl.acm.org

Grad-CAM: Why did you say that?

Interpretable explanations of black boxes by meaningful perturbation
RC. Fong, A. Vedaldi - Proceedings of the IEEE International ..., 2017 - openaccess.thecvf.com

sensitivity analysis

LRP, LIME, GradCam, Perturbation...

Brief Recent History of Post-Hoc XAI

Explaining nonlinear classification decisions with deep taylor decomposition
G Montavon, S Lapuschkin, A Binder, W Samek... - Pattern Recognition, 2017 - Elsevier

A unified approach to interpreting model predictions
SM Lundberg, SL Lee - Advances in neural information processing ..., 2017 - papers.nips.cc

Explaining recurrent neural network predictions in sentiment analysis

XAI for LSTMs
Theoretical frameworks for XAI
LRP, LIME,
GradCam, Perturbation ...

sensitivity analysis


W. Samek: Explainable AI: Concepts, Methods and Applications
Brief Recent History of Post-Hoc XAI

**XAI for transformers**: better explanations through conservative propagation

From "Where" to "What": Towards Human-Understandable Explanations through Concept Relevance Propagation

Beyond Explaining: Opportunities and Challenges of XAI-Based Model Improvement

---

**XAI for LSTMs**

Theoretical frameworks for XAI

**XAI for GNNs, Transformers ..,**

Concept-level XAI

**Beyond explaining**

---

sensitivity analysis

Linear Models

First step: Compute the prediction

\[ f(x) = w^T x \]
\[ = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d \]

Second step: Extract an explanation

\[ R_1 \leftarrow w_1 x_1 \]
\[ R_2 \leftarrow w_2 x_2 \]
\[ \vdots \]
\[ R_d \leftarrow w_d x_d \]

\[ R \leftarrow (R_1, R_2, \ldots, R_d) \]

Note: Feature i is regarded relevant if (1) it is present in the data and (2) it is used by the model.
Linear Models

First step: Compute the prediction

\[ f(x) = w^T x \]

\[ = w_1 x_1 + \cdots + w_d x_d \]

Second step:

\[ R_1 \leftarrow w_1 x_1 \]

\[ R_2 \leftarrow w_2 x_2 \]

\[ \vdots \]

\[ R_d \leftarrow w_d x_d \]

\[ \mathbf{R} \leftarrow (R_1, R_2, \ldots, R_d) \]

General Principle: "Explanation by Decomposition"

\[ f(x) \approx \sum_{i=1}^{d} R_i \]

Note: Feature i is regarded relevant if (1) it is present in the data and (2) it is used by the model.
Explanation Methods

**Perturbation-Based**
- Occlusion-Based (Zeiler & Fergus 14)
- Meaningful Perturbations (Fong & Vedaldi 17)
  ...

**Gradient-Based**
- Sensitivity Analysis (Simonyan et al. 14)
- (Simple) Taylor Expansions
- Gradient x Input (Shrikumar et al. 16)
  ...

**Surrogate-/Sampling-Based**
- LIME (Ribeiro et al. 16)
- SmoothGrad (Smilkov et al. 16)
  ...

**Propagation-Based**
- LRP (Bach et al. 15)
- Deep Taylor Decomposition (Montavon et al. 17)
- Excitation Backprop (Zhang et al. 16)
Perturbation-Based

**Idea:** Assess features relevance by testing the model response to their removal or perturbation.

\[
R_i = f(x) - f(x_{-i})
\]
**Perturbation-Based**

**Idea:** Assess features relevance by testing the model response to their removal or perturbation.

**Disadvantages**
- slow
- assumes locality
- perturbation may introduce artefacts

$\rightarrow$ unreliable

(e.g., occlusion in Zeiler & Fergus 2014)
Consider a sequence of inputs $x^{(0)}, x^{(1)}, \ldots, x^{(N)}$ interpolating between $x^{(0)} = 0$ and $x^{(N)} = x$.

Perform for each $n$ the perturbation analysis

$$R_i^{(n)} = f(x^{(n)}) - f(x_{-i}^{(n)})$$

where

$$x_{-i}^{(n)} = (x_1^{(n)}, \ldots, x_{i-1}^{(n)}, x_i^{(n-1)}, x_{i+1}^{(n)}, \ldots, x_d^{(n)})$$

Sum them up:

$$R_i = \sum_{n=1}^{N} R_i^{(n)}$$
**Integrated Gradients**

- **Observation:** When the interpolation steps are small enough and when $f$ is differentiable,

\[
R_i^{(n)} \approx [\nabla f(x^{(n)})]_i \cdot (x_i^{(n)} - x_i^{(n-1)})
\]

where the function’s gradient appears.

- At each step, the perturbation for all dimensions can be computed using only one gradient evaluation.

- This is the integrated gradients method (in discretized form) (Sundararajan et al. 2017)
Integrated Gradients

- **Integrated Gradients (IG)** (Sundararajan et al. 2017):

\[ R_i = \sum_{n=1}^{N} [\nabla f(x^{(n)})]_i \cdot (x^{(n)}_i - x^{(n-1)}_i) \]

- **Gradient × Input (GI)** (Shrikumar et al. 2016)

\[ R_i = [\nabla f(x)]_i \cdot x_i \]

i.e. an input feature \( i \) contributes if it is present in the data \((x_i > 0)\) and if the model reacts to it \( ([\nabla f(x)]_i > 0)\).

**Proposition:** When \( x^{(0)}, x^{(1)}, \ldots, x^{(N)} \) linearly interpolate between \( x^{(0)} = 0 \) and \( x^{(N)} = x \), and when \( f \) is positively homogeneous, i.e. \( \forall t \geq 0 : f(tx) = tf(x) \), then IG and GI produce the same result.
Problem: Gradients are 'Shattered'

- We look at the DNN output (and its gradient) along some trajectory in the input space, e.g. an athlete lifting a barebell.
- The function is relatively stable, but the gradient strongly oscillates and appears noisy (cf. Balduzzis et al. 2017)
Shattered Gradients: A Construction

Consider the function:

\[ g(x) = 2 \cdot \text{ReLU}(x) - 4 \cdot \text{ReLU}(x - 0.5) \]

defined on the interval \([0, 1]\).

We apply the function recursively to form a deep neural network.

<table>
<thead>
<tr>
<th>function</th>
<th>output</th>
<th>max slope</th>
<th># linear pieces</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g(x) )</td>
<td>([0, 1])</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( g \circ g(x) )</td>
<td>([0, 1])</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>( g \circ g \circ g(x) )</td>
<td>([0, 1])</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>( g \circ g \circ g \circ g(x) )</td>
<td>([0, 1])</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

Potentially exponential growth of gradient and linear pieces (cf. Montufar et al. 2014)
SmoothGrad: "Removing Noise by Adding Noise"

**Idea:** Perform the gradient-based analysis with multiple random perturbations $\epsilon_1, \ldots, \epsilon_T$ of the input, and average the explanations. (Smilkov et al. 2017)

**Example:** Smooth Gradient $\times$ Input

$$R_i = \frac{1}{T} \sum_{t=1}^{T} [\nabla f(x + \epsilon_t)]_i [x + \epsilon_t]_i$$
SmoothGrad

Advantages

- Reduces explanation noise.
- Simple to implement (just call the same code multiple time)
- Widely applicable (can be applied on top of any explanation technique).

Limitations

- Computation cost increases by a factor $T$ while explanation noise is in the best case only reduced by a factor $\sqrt{T}$.
- Adding noise to the input implies that we explain a slightly different quantity than the input (this may add a bias to the explanation).
From Function-Based to Propagation-Based

Questions:

- Can using the structure of the network *explicitly* (e.g. by running a special propagation pass) help to produce a better explanation?

- Can this approach reduce explanation noise *without* having to evaluate the function multiple times?
The 'Deconvolution' Method

- Max-pooling layers: propagate to the winner
- Convolutional layers: convolve with transposed weights
- ReLU layers: apply the ReLU function

Image source:
Zeiler et al. (2014) Visualizing and Understanding Convolutional Networks

(Zeiler & Fergus 2014)
The 'Deconvolution' Method

- **Observation:** Gradient noise has disappeared $\Rightarrow$ leveraging structure is useful.

- **Limitation:** The method was meant as a visualization rather than as an explanation (it does not tell how much each input variable has contributed to the prediction).

Image source:
Zeiler et al. (2014) Visualizing and Understanding Convolutional Networks
Layer-wise Relevance Propagation

**Ideas:**

- Use the structure of the neural network to robustly compute relevance scores for the input features.
- Propagate the output of the network backwards by means of propagation rules.
- Propagation rules can be tuned for explanation quality. E.g. sensitive in top-layers, robust in lower layers.

\[ R_j = \sum_k \frac{a_j w_{jk}}{\varepsilon_k + \sum_{0,j} a_j w_{jk}} R_k \]

(Bach et al. 2015)
Layer-wise Relevance Propagation

Idea: Redistribute the evidence for class rooster back to image space.
Layer-wise Relevance Propagation

Theoretical Interpretation:
Deep Taylor Decomposition

Simple LRP rule (Bach et al. 2015)

\[
R_i^{(l)} = \sum_j \frac{a_i \cdot w_{ij}}{\sum_i a_{i'} \cdot w_{i'j}} R_j^{(l+1)}
\]

Every neuron gets its "share" of the redistributed relevance.
Layer-wise Relevance Propagation

Explanation

Theoretical Interpretation: Deep Taylor Decomposition

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 \( \alpha + \beta = 1 \)
Layer-wise Relevance Propagation

Explanation

Layer-wise relevance conservation

$$\sum_i R_i = \ldots = \sum_i R_i^{(l)} = \sum_j R_j^{(l+1)} = \ldots = f(x)$$
Dissecting a LRP Propagation Rule

**Example:** LRP-\(\gamma\) (Montavon et al. 2019)

\[
R_j = \frac{\sum_k a_j(w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j(w_{jk} + \gamma w_{jk}^+)} R_k
\]

- \(a_j(w_{jk} + \gamma w_{jk}^+):\) Contribution of neuron \(a_j\) to the activation \(a_k\).
- \(R_k\) ‘Relevance’ of neuron \(k\) available for redistribution.
- \(\sum_{0,j} a_j(w_{jk} + \gamma w_{jk}^+)\) Normalization term that implements conservation.
- \(\sum_k:\) Pool all ‘relevance’ received by neuron \(j\) from the layer above.

**Note:** The parameter \(\gamma\) controls by how much positive contributions are favored. As \(\gamma\) increases, negative contributions start to disappear.
Dissecting a LRP Propagation Rule

**Example:** LRP-\(\gamma\) (Montavon et al. 2019)

\[
R_j = a_j \cdot \left( \sum_k \frac{(w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k \right)
\]

- \(a_j\): Activation of neuron \(j\).
- \(\sum_k \ldots\): Sensitivity of neural network output to \(a_j\).

i.e. similar interpretation as for Gradient × Input, but now at each layer.
Effect of LRP Rules on Explanation

LRP rules must be chosen carefully to deliver best explanation quality. Generally, LRP rules are set different at each layer (Montavon et al. 2019).
Layer-wise Relevance Propagation

Advantages

- Good explanation quality on deep networks.
- Fast (in the order of a single forward/backward pass).
- Flexible (the multiple hyperparameters can be tuned to match the user needs).

Limitations

- The LRP propagation strategy must be adapted to each new architecture.
- LRP makes some assumptions about the structure of the model (i.e. it works for many neural networks but not for all models).
Connections between Explanation Methods

- Perturbation-based explanations
  - IntGrad
  - SmoothGrad
  - Perturbations (continuous, reduction, + robustness)

- Propagation-based explanations
  - Deconvolution
  - LRP

+ robustness
More Explanation Methods

Other methods that have been proposed to attribute the prediction to input features:

- **LIME [12]**: learns a local surrogate model and analyze it.
- **SHAP [8]**: based on the game theory framework of Shapley values.
- **Meaningful Perturbations [7]**: synthesizes an optimal perturbation with gradient ascent.
- **Grad-CAM [14]**: combines gradient-based and propagation-based approaches.

Gradient-Based: Sensitivity Analysis

- **Image**: Explains what influences prediction “cars”.
- **Sensitivity Analysis**: Explains prediction “cars” as is.
- **LRP / Deep Taylor**: Slope decomposition
  \[ \sum_i R_i = \| \nabla_x f \|^2 \]
  Value decomposition
  \[ \sum_i R_i = f(x) \]

(Montavon et al. 2017)
Surrogate-Based: LIME

**Idea:** Fit simple surrogate model

\[ err = \sum_i (f(x_i) - w^T x_i)^2 \]

(Ribeiro et al. 2016)
## Comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Examples</th>
<th>Agnostic</th>
<th>Efficient</th>
<th>Determ.</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pert.-Based</td>
<td>[ZF14, Sha53, FV17]</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>OOD</td>
</tr>
<tr>
<td>Grad.-Based</td>
<td>[BSH+10, SVZ14, STY17]</td>
<td>PARTLY</td>
<td>YES</td>
<td>YES</td>
<td>Shattering</td>
</tr>
<tr>
<td>Sur.-Based</td>
<td>[RSG16, RSG18]</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>Surrogate</td>
</tr>
<tr>
<td>Prop.-Based</td>
<td>[BBM+15, MLB+17b, ZBL+18]</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>Rules*</td>
</tr>
</tbody>
</table>

*Deep Taylor Decomposition [MLB+17b] offers a theoretical framework to design these rules.*
Implementation & Codes
Implementation

Implementation of different techniques can be made simple by using special techniques or tricks:

- Gradient × Input
  - Automatic differentiation
- Deconvolution
  - Backward hooks
- Layer-wise relevance propagation
  - .detach()
Implementing Gradient x Input

Load VGG-16 Model

```
In [2]: import torchvision
model = torchvision.models.vgg16(pretrained=True)
model.eval();
```

Prepare to compute input gradient

```
In [3]: X.grad = None
X.requires_grad_(True);
```

Compute explanation: $R_i = [\nabla f(\mathbf{x})]_i \cdot x_i$

```
In [4]: model.forward(X)[0,483].backward()
R = (X*X.grad)
```

Visualize explanation

```
In [5]: utils.heatmap(R[0].sum(dim=0),'explanation-gi.png')
```
Implementing Deconvolution (guided version, \( x \) input)

Build a hook that rectifies the gradient

In [6]:
```python
def hook(mod, grad_in, grad_out):
    return (grad_in[0].clamp(min=0),)
```

Register this hook in ReLU layers

In [7]:
```python
for i in [1,3,6,8,11,13,15,18,20,22,25,27,29]:
    model.features[i].register_backward_hook(hook)
```

Apply Gradient \( x \) Input

In [8]:
```python
X.grad = None
X.requires_grad_(True);
model.forward(X)[0,483].backward()
R = (X*X.grad)
utils.heatmap(R[0].sum(dim=0),"explanation-grad.png")
```
Implementing LRP

**Observation:** Writing relevance scores as \( R_j = a_j c_j \) and \( R_k = a_k c_k \), the LRP-\( \gamma \) propagation rule can also be expressed as:

\[
c_j = \sum_k (w_{jk} + \gamma w_{jk}^+) \frac{a_k}{p_k} c_k \quad \text{with} \quad p_k = \sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)
\]

and this can be further simplified to

\[
c_j = \sum_k \frac{\partial p_k}{\partial a_j} \frac{a_k}{p_k} c_k = \sum_k \frac{\partial}{\partial a_j} \left( p_k \cdot \left[ \frac{a_k}{p_k} \right]_{\text{cst.}} \right) c_k
\]

which has the structure of the multivariate chain rule for gradient propagation.

Now, we can replace \( a_k \) by \( p_k \cdot \left[ a_k / p_k \right]_{\text{cst.}} \) in the forward pass and then run standard automatic differentiation get the LRP explanation (Montavon et al. 2019).
Implementing LRP (simplified)

Build an equivalent forward pass where part of it is detached

```python
In [11]:

class Conv(torch.nn.Module):
    
def __init__(self, conv, gamma):
        torch.nn.Module.__init__(self)
        self.conv = conv
        self.pconv = copy.deepcopy(conv)
        self.pconv.weight = torch.nn.Parameter(
            conv.weight+gamma*conv.weight.clamp(min=0)
        )
    
def forward(self, X):
        z = self.conv.forward(X)
        zp = self.pconv.forward(X)
        return zp * (z / zp).data
```
Implementing LRP (simplified)

Replace layers by modified layers

\[
\begin{align*}
\text{In [12]}: & \quad f = model.features \\
& \quad \text{for } i \text{ in [2]}: \quad f[i] = \text{Conv}(f[i], 1) \\
& \quad \text{for } i \text{ in [5, 7]}: \quad f[i] = \text{Conv}(f[i], 0.3) \\
& \quad \text{for } i \text{ in [10, 12, 14]}: \quad f[i] = \text{Conv}(f[i], 0.1) \\
& \quad \text{for } i \text{ in [17, 19, 21]}: \quad f[i] = \text{Conv}(f[i], 0.03) \\
& \quad \text{for } i \text{ in [24, 26, 28]}: \quad f[i] = \text{Conv}(f[i], 0.01)
\end{align*}
\]

Apply Gradient × Input

\[
\begin{align*}
\text{In [13]}: & \quad X.grad = \text{None} \\
& \quad X.requires_grad(True) \\
& \quad \text{model.forward}(X)[0, 483].backward() \\
& \quad R = (X*\text{X.grad}) \\
& \quad \text{utils.heatmap}(R[0].sum(dim=3), 'explanation-lrp.png')
\end{align*}
\]
Zennit Toolbox

Zennit registers hooks at Pytorch's Module level, to modify the backward pass to produce rule-based attributions like LRP.

(Anders et al. 2021)

Zennit (Zennit explains neural networks in torch) is a high-level framework in Python using Pytorch for explaining/exploring neural networks.

https://github.com/chr5tphr/zennit
Zennit Toolbox

Composites (zennit/composites.py) are a way of choosing the right hook for the right layer. In addition to the abstract NameMapComposite, which assigns hooks to layers by name, and LayerMapComposite, which assigns hooks to layers based on their Type, there exist explicit Composites, some of which are EpsilonGammaBox (ZBox in input, Epsilon in dense, Gamma in convolutions) or EpsilonPlus (Epsilon in dense, ZPlus in convolutions). All composites may be used by directly importing from zennit.composites, or by using their snake-case name as key for zennit.composites.COMPOSITES.

Canonizers (zennit/canonziers.py) temporarily transform models into a canonical form, if required, like SequentialMergeBatchNorm, which automatically detects and merges BatchNorm layers followed by linear layers in sequential networks, or AttributeCanonizer, which temporarily overwrites attributes of applicable modules, e.g. to handle the residual connection in ResNet-Bottleneck modules.

Attributors (zennit/attribution.py) directly execute the necessary steps to apply certain attribution methods, like the simple Gradient, SmoothGrad or Occlusion. An optional Composite may be passed, which will be applied during the Attributor's execution to compute the modified gradient, or hybrid methods.
Zennit Toolbox

```python
import torch
from torchvision.models import vgg16_bn
from zennit.composites import EpsilonGammaBox
from zennit.canonizers import SequentialMergeBatchNorm
from zennit.attractions import Gradient

data = torch.randn(1, 3, 224, 224)
model = vgg16_bn()

canonizers = [SequentialMergeBatchNorm()]
composite = EpsilonGammaBox(low=-3., high=3., canonizers=canonizers)

with Gradient(model=model, composite=composite) as attributor:
    out, relevance = attributor(data, torch.eye(1000)[[0]])
```

A similar setup using the example script produces the following attribution heatmaps:
References
From XAI to XXAI

(2019)  (2022)

W. Samek: Explainable AI: Concepts, Methods and Applications
W Samek, G Montavon, S Lapuschkin, C Anders, KR Müller

Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications

Proceedings of the IEEE, 109(3):247-278, 2021

With the broader and highly successful usage of machine learning (ML) in industry and the sciences, there has been a growing demand for explainable artificial intelligence (XAI). Interpretability and explanation methods for gaining a better understanding of the problem-solving abilities and strategies of nonlinear ML, in particular, deep neural networks, are, therefore, receiving increased attention. In this work, we aim to: 1) provide a timely overview of this active emerging field, with a focus on "post hoc" explanations, and explain its theoretical foundations; 2) put interpretability algorithms to a test both from a theory and comparative evaluation perspective using extensive simulations; 3) outline best practice aspects, i.e., how to best include interpretation methods into the standard usage of ML; and 4) demonstrate successful usage of XAI in a representative selection of application scenarios. Finally, we discuss challenges and possible future directions of this exciting foundational field of ML.
References

Tutorial / Overview Papers


References

Methods Papers


References

Explaining Beyond DNN Classifiers


References

Concept-Level Explanations


Evaluation of Explanations

References

Model Validation and Improvement

  Nature Communications, 10:1096, 2019 [preprint, bibtex]

  Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2912-2920, 2016 [preprint, bibtex]

  Information Fusion, 77:261-295, 2022 [preprint, bibtex]


- L Weber, S Lapuschkin, A Binder, W Samek. Beyond Explaining: Opportunities and Challenges of XAI-Based Model Improvement
  arXiv:2203.08008, 2022 [preprint], bibtex

- J Sun, S Lapuschkin, W Samek, A Binder. Explain and Improve: LRP-Inference Fine Tuning for Image Captioning Models
  Information Fusion, 77:233-246, 2022 [preprint], bibtex

  arXiv:2202.03482, 2022 [preprint], bibtex

- J Sun, S Lapuschkin, W Samek, Y Zhao, NM Cheung, A Binder. Explanation-Guided Training for Cross-Domain Few-Shot Classification
  Proceedings of the 25th International Conference on Pattern Recognition (ICPR), 7609-7616, 2021 [preprint, bibtex]
References

Software Papers


References

Application to Sciences

- I Sturm, S Bach, W Samek, KR Müller. **Interpretable Deep Neural Networks for Single-Trial EEG Classification**
  Journal of Neuroscience Methods, 274:141–145, 2018 [preprint, bibtex]

  Scientific Reports, 10:6423, 2020 [preprint, bibtex]


- F Horst, S Lapuschkin, W Samek, KR Müller, WI Schöllhorn. **Explaining the Unique Nature of Individual Gait Patterns with Deep Learning**
  Scientific Reports, 9:2391, 2019 [preprint, bibtex]

  ACM Transactions on Computing for Healthcare, 3(2):1-21, 2022 [preprint], bibtex


- P Seeberger, A Binder, R Saitenmacher, M Bockmayr, M Alber, P Jurmeister, F Klauschen, KR Müller. **Interpretable Deep Neural Network to Predict Estrogen Receptor Status from Haematoxylin-Eosin Images**
  Artificial Intelligence and Machine Learning for Digital Pathology, Springer LNCS, 12090, 16-37, 2020 [bibtex]

  NeuroImage, 2022 [bibtex]

- P Keyl, M Bockmayr, D Heim, G Dembach, G Montavon, KR Müller, F Klauschen. **Patient-level proteomic network prediction by explainable artificial intelligence**
  NPJ Precis Oncol. 6(1):35, 2022 [bibtex]
References

Application to Text


References

Application to Images & Faces


References

Application to Video


Application to Speech

References

Application to Neural Network Pruning


- D Becking, M Dreyer, W Samek, K Müller, S Lapuschkin. ECQx: Explainability-Driven Quantization for Low-Bit and Sparse DNNs xxAI - Beyond Explainable AI, Springer LNAI, 13200:271-296, 2022 [preprint, bibtex]


Interpretability and Causality

Thank you for your attention

http://www.heatmaping.org

- Tutorials
- Software
- Online Demos

Interested in PhD / PostDoc in Berlin?

Please contact me:
wojciech.samek@hh.fraunhofer.de