Part 2: Methods for Explaining DNNs

Wojciech Samek, Grégoire Montavon

September 18, 2020
Outline of Part 2

- Defining the problem of explanation
- Self-explainable models
  - Advantages & limitations
- Post-hoc explanations
  - Perturbation-based approaches
  - Propagation-based approaches
Consider we have a trained model \( f \).

We give to this model a data point \( \mathbf{x} \in \mathbb{R}^d \), where each feature \( x_i \) composing it is assumed to be interpretable (e.g. physical measurement, pixel, or word).

The model produces for \( \mathbf{x} \) an output \( f(\mathbf{x}) \).

We would like to build an explanation \( \mathbf{R} = (R_i)_i \) indicating to what extent each feature \( i \) contributes to the prediction.
Illustration for a Linear Model

- **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

  \[ R_1 \leftarrow w_1 x_1 \]
  \[ R_2 \leftarrow w_2 x_2 \]
  \[ \vdots \]
  \[ R_d \leftarrow w_d x_d \]
  \[ \mathbf{R} \leftarrow (R_1, R_2, \ldots, R_d) \]
**Question:** How to trace which input features have contributed to the prediction in a more general deep model?
Self-Explainable Deep Networks

Idea: Restrict connectivity to ease the problem of attribution.
The Generalized Additive Model (GAM) [6]

Observation: Attribution is easy: $R_1 = g_1(x_1), R_{23} = g_{23}(x_2, x_3)$ \ldots
Bag-Of-Local-Features [5]

Image source: Brendel et al. (2019) Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet
Bag-Of-Local-Features [5]

With a larger receptive field (i.e. with less restrictions on the model), the prediction accuracy improves but the explanation becomes more blurry.
Advantages and Limitations of Self-Explainable Models

Advantages

- Explanations can be **easily** extracted without further analysis.
- The model can be designed to be **maximally interpretable** (e.g. by penalizing the use of uninterpretable features).
- Model constraints can be **relaxed** when explanation is coarse-grained (e.g. pixels → patches).

Limitations

- Self-explainable model might **lack representation power**, e.g. the GAM cannot represent a simple max-pooling operation.
- Even when the model predicts well ...
  - The model’s strategy may be influenced by its restricted structure, and this may lead to a **less natural** prediction strategy from which it is harder to extract knowledge.
  - The model’s strategy will likely be **computationally less efficient** than a standard model.
Beyond Generalized Additive Models

Prediction

detect complex objects in images

trade spatial resolution for semantic resolution

output

input
Example: Convolutional Neural Networks

Properties

- Top-layers can capture long-range interactions.
- Increasingly many features can be built in higher layers.
- Representation remains finite-dimensional at each layer (→ computationally efficient).
Explaining Beyond Generalized Additive Models

Prediction

- detect complex objects in images
- trade spatial resolution for semantic resolution

Explanation

- spatial resolution too low for explanation
- how to redistribute to pixels?
A Different Approach to Explanation: Perturbation

Examples from the literature:

- Occlusion [18], Prediction Difference Analysis [19]
Perturbation Analysis

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

Advantages

▶ Can be applied to any function $f(x)$.
▶ Consistent for GAMs ($R_i = f(x) - f(x_{-i}) = g_i(x)$).

Limitations

▶ Slow (function $f$ must be reevaluated for each occlusion)
▶ Intrinsically local, e.g. fails to explain max-pooling when several features in the pool are activated.
▶ Potentially biased by what is inserted in place of the removed patch. (Alternative: remove and inpaint [1, 13].)
Continuous Perturbations

- 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

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

- Sum them up:

$$R_i = \sum_{n=1}^{N} R_i^{(n)}$$
Continuous Perturbations

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

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

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) [17].
Integrated Gradients and Gradient $\times$ Input

- Integrated Gradients (IG) [17]:
  
  $$R_i = \sum_{n=1}^{N} [\nabla f(x^{(n)})]_i \cdot (x_i^{(n)} - x_i^{(n-1)})$$

- Gradient $\times$ Input (GI) [15, 2, 9]:
  
  $$R_i = [\nabla f(x)]_i \cdot x_i$$

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.
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.

Proof: We start with IG and arrive at GI using a property of positively homogeneous functions (cf. note).

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

(1)

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

(2)

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

(3)

Note: A positively homogeneous function satisfies \( \forall t \geq 0 : f(tx) = tf(x) \). Differentiating on both sides gives

\[
\frac{\partial}{\partial x} f(tx) = \frac{\partial}{\partial x} tf(x)
\]

\[
t \nabla f(tx) = t \nabla f(x)
\]

therefore, the gradient is the same on any point on the segment \((0, x)\).
Gradient × Input in Practice

Example: Gradient × Input explanation of the VGG-16 neural network output neuron ‘viaduct’ for a given input image:

Observation: There is an exceedingly large amount of positive (red) and negative (blue) scores. Explanations also appear noisy and are hard to interpret.
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. [4]).
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. [11]).
SmoothGrad [16]: “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.

**Example:** Smooth Gradient × Input

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

---

input

T = 1  T = 4  T = 16  T = 64

noise = 0.5 std  noise = 1.0 std
SmoothGrad

**Advantages**

- Reduces explanation noise.
- Simple to implement (just call the same code multiple times).
- 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).

![Images of SmoothGrad results for different $T$ values and noise levels]
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 [18]

- 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
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 (LRP) [3, 10]

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.
Layer-wise Relevance Propagation (LRP) [3, 10]

Some notation:

- $j$ and $k$: neurons from successive layers
- $w_{jk}$: weight connecting neuron $j$ to neuron $k$
- $w_{0k}$: bias for neuron $k$.
- $\sum_{0,j}$ sum over all input neurons $j$ of neuron $k$ and the bias.
- ReLU neuron: $a_k = \max(0, \sum_{0,j} a_j w_{jk})$. 

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

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

Example: LRP-\(\gamma\) [10]

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

- \(a_j(w_{jk} + \gamma w_{jk}^+)\): 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.
Dissecting a LRP Propagation Rule (2nd view)

**Example:** LRP-\(\gamma\) [10]

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

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

i.e. similar interpretation as for Gradient \(\times\) 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 (cf. [10] for heuristics).
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

self-explainable models

perturbation-based explanations

propagation-based explanations

GAMs

IntGrad

Grad x Input

SmoothGrad

Deconvolution

LRP

continuous

reduction

+ robustness

+ robustness

in part 3

Bag-of-local-features

self-explainable models

perturbation-based explanations

propagation-based explanations

GAMs

IntGrad

Grad x Input

SmoothGrad

Deconvolution

LRP

continuous

reduction

+ robustness

+ robustness

in part 3

Bag-of-local-features

self-explainable models

perturbation-based explanations

propagation-based explanations

GAMs

IntGrad

Grad x Input

SmoothGrad

Deconvolution

LRP

continuous

reduction

+ robustness

+ robustness

in part 3

Bag-of-local-features

self-explainable models

perturbation-based explanations

propagation-based explanations

GAMs

IntGrad

Grad x Input

SmoothGrad

Deconvolution

LRP

continuous

reduction

+ robustness

+ robustness

in part 3

Bag-of-local-features

self-explainable models

perturbation-based explanations

propagation-based explanations

GAMs

IntGrad

Grad x Input

SmoothGrad

Deconvolution

LRP

continuous

reduction

+ robustness

+ robustness

in part 3

Bag-of-local-features

self-explainable models

perturbation-based explanations

propagation-based explanations

GAMs

IntGrad

Grad x Input

SmoothGrad

Deconvolution

LRP

continuous

reduction

+ robustness

+ robustness

in part 3

Bag-of-local-features
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.
Self-explainable models can be practical, but they often lack sufficient representation power and can be computationally costly.

Explaining general DNNs is hard (no directly identifiable contributions, gradient noise), but possible.

Two important categories of ‘post-hoc’ explanation techniques (perturbation-based and propagation-based).

The LRP explanation technique is specially designed to explain deep networks (perform attribution by taking advantage of the layered structure).
References I

Removing input features via a generative model to explain their attributions to classifier’s decisions.

Gradient-based attribution methods.

On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation.

The shattered gradients problem: If resnets are the answer, then what is the question?

Approximating cnns with bag-of-local-features models works surprisingly well on imagenet.


References III

On the number of linear regions of deep neural networks.
In NIPS, pages 2924–2932, 2014.

"why should I trust you?": Explaining the predictions of any classifier.

Toward interpretable machine learning: Transparent deep neural networks and beyond.


Not just a black box: Learning important features through propagating activation differences.
Smoothgrad: removing noise by adding noise.

Axiomatic attribution for deep networks.

Visualizing and understanding convolutional networks.

Visualizing deep neural network decisions: Prediction difference analysis.
In *ICLR (Poster)*. OpenReview.net, 2017.