Optimizing Explanations by Network Canonization and Hyperparameter Search

Frederik Pahde\textsuperscript{1} Galip Ümit Yolcu\textsuperscript{1,2} Alexander Binder\textsuperscript{3,4} Wojciech Samek\textsuperscript{1,2,5,*} Sebastian Lapuschkin\textsuperscript{1,*}

\textsuperscript{1}Department of Artificial Intelligence, Fraunhofer Heinrich-Hertz-Institute, Germany
\textsuperscript{2}Technische Universität Berlin, Germany
\textsuperscript{3}ICT Cluster, Singapore Institute of Technology, Singapore
\textsuperscript{4}University of Oslo, Norway
\textsuperscript{5}BIFOLD – Berlin Institute for the Foundations of Learning and Data, Germany

* corresponding authors \{wojciech.samek|sebastian.lapuschkin\}@hhi.fraunhofer.de

Abstract

Explainable AI (XAI) is slowly becoming a key component for many AI applications. Rule-based and modified backpropagation XAI approaches however often face challenges when being applied to modern model architectures including innovative layer building blocks, which is caused by two reasons. Firstly, the high flexibility of rule-based XAI methods leads to numerous potential parameterizations. Secondly, many XAI methods break the implementation-invariance axiom because they struggle with certain model components, e.g., BatchNorm layers. To address that issue, a method has been suggested, a method that fuses BN layers into neighboring linear layers without changing the underlying function of the model \cite{2015arXiv151106802F}. More-\textsuperscript{over}, there is only little quantifiable evidence that model canonization is beneficial for XAI. In this work, we propose canonizations for currently relevant model blocks applicable to popular deep neural network architectures, including VGG, ResNet, EfficientNet, DenseNets, as well as Relation Networks. We further suggest a XAI evaluation framework with which we quantify and compare the effects of model canonization for various XAI methods in image classification tasks on the Pascal VOC and ILSVRC2017 datasets, as well as for Visual Question Answering using CLEVR-XAI. Moreover, addressing the former issue outlined above, we demonstrate how our evaluation framework can be applied to perform hyperparameter search for XAI methods to optimize the quality of explanations. Code is available on \url{https://github.com/frederikpahde/xai-canonization}.

1. Introduction

In recent years, Machine Learning (ML) has been increasingly applied to high-stakes decision processes with a huge impact on human lives, such as medical applications \cite{2019arXiv191101006L, 2019arXiv191101006L}, credit scoring \cite{2017arXiv170201035S}, criminal justice \cite{2015arXiv151200589H}, and hiring decisions \cite{2019arXiv191100769C}. Therefore, awareness has been raised for the need of neural networks and their predictions to be transparent and explainable \cite{2019arXiv191100577H}, which makes Explainable AI (XAI) a key component of modern ML systems. Rule-based and modified backpropagation-based XAI methods, such as DeepLift \cite{2019arXiv191101006L}, Layer-wise Relevance Propagation (LRP) \cite{2015arXiv151106802F}, and Excitation Backprop \cite{2019arXiv191100577H}, that are among the most prominent XAI approaches due to their high faithfulness and efficiency, however, struggle when being applied to modern model architectures with innovative building blocks. This is caused by two problems: Firstly, rule-based XAI methods provide large flexibility thanks to configurable rules which can be tailored to the model architecture at hand. This comes at the cost of numerous potential XAI method parameterizations, particularly for complex model architectures. However, finding optimal parameters is barely researched and often neglected, which can cause these methods to yield suboptimal explanations. Secondly, earlier works \cite{2019arXiv191100769C} have shown that many XAI methods break implementation invariance, which has been defined as an axiom for explanations \cite{2019arXiv191100769C}. This is caused by certain layer types for which no explanation rules have been defined yet, e.g., BatchNorm (BN) layers. To address this issue, model canonization has been suggested, a method that fuses BN layers into neighboring linear layers without changing the underlying function of the model \cite{2015arXiv151106802F, 2015arXiv150202550B}, arguably leading to improved explanations for simple model architectures (VGG, ResNet) \cite{2019arXiv191100769C}. However, what constitutes a “good” explanation is only vaguely defined and many, partly contradicting, metrics for the quality of expla-
nations have been proposed. Therefore, tuning hyperparameters of XAI methods and measuring the benefits of model canonization for XAI are non-trivial tasks.

To that end, we propose an evaluation framework, in which we evaluate XAI methods w.r.t their faithfulness, complexity, robustness, localization capabilities, and behavior with regard to randomized logits, following the authors of [17]. We apply our framework to (1) measure the impact of canonization and (2) demonstrate how hyperparameter search can improve the quality of explanations. Therefore, we first extend the model canonization approach to modern model architectures with high interconnectivity, e.g., DenseNet variants. We apply our evaluation framework to measure the benefits of model canonization for various image classification model architectures (VGG, ResNet, EfficientNet, DenseNet) using the ILSVRC2017 and Pascal VOC 2012 datasets, as well as for Visual Question Answering (VQA) with Relation Networks using the CLEVR-XAI dataset [4]. We show that generally model canonization is beneficial for all tested architectures, but depending on which aspect of explanation quality is measured, the impact of model canonization differs. Moreover, we demonstrate how our XAI evaluation framework can be leveraged for hyperparameter search to optimize the explanation quality from different points of view.

2. Related Work

2.1. XAI Methods

XAI methods can broadly be categorized into local and global explanations. While local explainers focus on explaining the model decisions on specific inputs, global explanation methods aim to explain the model behavior in general, e.g., by visualizing learned representations. For image classification tasks, local XAI methods assign relevance scores to each input unit, expressing how influential that unit (e.g., an input pixel) has been for the inference process. Many XAI methods are (modified) backpropagation approaches. To compute the importance of features in the detection of a certain class, they start from the output of the network, backpropagating importance values layer by layer, depending on the parameters and/or hidden activations of each layer. Saliency maps [6, 28, 38] are generated by computing the gradient \( \frac{\partial f(x)}{\partial x} \), where \( f(x) \) is the model’s prediction for an input sample \( x \). This yields a feature map, where each value indicates the model’s sensitivity towards the corresponding feature. Guided Backpropagation [41] also uses the gradients, but applies the ReLU function to computed gradients in ReLU activation layers in the backpropagation pass. This filters out the flow of negative information, allowing to focus on the parts of the image where the desired class is detected. Integrated Gradients [42] accumulates the activation gradients on a straight path in the input space, starting from a baseline image \( x' \) selected beforehand, to the datapoint of interest. Formally, the attribution to the \( i \)-th feature is given by \( \int_{\rho=0}^{1} \frac{\partial f(x + \rho(x' - x))}{\partial x_i} d\rho \).

SmoothGrad [40] aims to reduce the noise in saliency maps by sampling datapoints in the neighborhood of the original datapoint, and taking the average saliency map. Since these XAI methods rely only on the gradients of the total function computed by the network, they are implementation invariant, meaning they produce the same explanations for different implementations of the same function.

LRP [5] operates by redistributing the relevance scores of neurons backwards up to the input features. More precisely, LRP distributes the activation of the output neuron of interest to the previous layers in a way that preserves relevance across layers. Several rules have been defined (e.g., LRP-\( \alpha \), LRP-\( \gamma \), LRP-\( \alpha \beta \)), which can be combined in meaningful configurations according to the types and positions of layers in the neural network. Excitation Backprop [48] is a backpropagation method that is equivalent to LRP-\( \alpha \beta \) [27], which has a probabilistic interpretation. DeepLIFT [37] is another rule-based method, where a reference image (e.g., the mean over the training population) is selected in addition. Using the associated rules, the differences in the activations of neurons on the reference image and the target image are backpropagated to the input. In addition to backpropagation-based XAI methods, there are also other approaches. Prominent examples are SHAP [24], which uses the game theoretic concept of Shapley values to find the contribution of each input feature to the model output, and LIME [31], which fits an interpretable model to the original model output around the given input. Both methods treat the model as a black box, only using outputs for certain inputs. As such, they are also implementation independent.

2.2. Evaluation of XAI Methods

While various XAI methods have been developed, the quantitative evaluation thereof is often neglected and explanations of XAI methods are often only compared by visually inspecting heatmaps. To address this issue, many XAI metrics have been introduced in recent years [1, 17]. However, there is no consensus on which metric to use and moreover, each metric evaluates explanations from different viewpoints, partly with contradictory objectives. Broadly speaking, XAI metrics can be categorized into five classes: Faithfulness measures metrics whether an explanation truly represents features used by the model. For instance, Pixel Flipping [5] measures the difference in output scores of the correct class, when replacing pixels in descending order of their relevance scores with a baseline value (e.g., black pixel or mean pixel). If the score decreases quickly, i.e., after replacing only a few highly relevant pixels, the explanation is considered as highly faithful. Region Perturbation [33] further generalizes Pixel Flipping by replacing input regions...
instead of single pixels. Faithfulness correlation [7] replaces a random subset of attribution with a baseline value and measures the correlation between the sum of attributions in the subset and the difference in model output. Robustness metrics measure the robustness of explanations towards small changes in the input. Prominent examples are Max-Sensitivity and Avg-Sensitivity [46], which use Monte Carlo sampling to measure the maximum and average sensitivity of an explanation for a given XAI method. Localization metrics measure how well an explanation localizes the object of interest for the underlying task. Consequently, in addition to the input sample and an explanation function, ground-truth localization annotations are required. Examples for localization metrics are Relevance Rank Accuracy (RRA) and Relevance Mass Accuracy (RMA) [4]. RRA measures the fraction of high-intensity relevances within the (binary) ground truth mask as \( RRA = \frac{P_{\text{top-K}} \cap \text{GT}}{|\text{GT}|} \), where GT is the ground truth, K is the size of the ground truth mask and \( P_{\text{top-K}} \) is the set of pixels sorted by relevance in decreasing order. Similarly, RMA measures the fraction of the total relevance mass within the ground truth mask and can be computed as \( \text{RMA} = \frac{R_{\text{within}}}{R_{\text{total}}} \) where \( R_{\text{within}} \) is the sum of relevance scores for pixels within the ground truth mask and \( R_{\text{total}} \) is the sum of all relevance scores. Complexity metrics measure how concise explanations are. For example, the authors of [11] use the Gini Index of the total attribution vector to measure its sparseness, while [7] propose an entropy-derived measure. Randomization metrics measure by how much explanations change when randomizing model components. For instance, the random logit test [39] measures the distance between the original explanation and the explanation with respect to a random other class.

2.3. Challenges of Rule-Based/Modified Backpropagation Methods

No Implementation Invariance: From a functional perspective, it is desirable for an XAI method to be implementation invariant, i.e., the explanations for predictions of two different neural networks implementing the same mathematical function should always be identical [42]. However, rule-based and modified backpropagation approaches explain predictions from a message-passing point of view, which, by design, is affected by the structure of the predictor. This is impressively demonstrated by Montavon et al. [25], where the authors compute explanations for two different implementations of the same mathematical function and the relevance scores differ tremendously. Therefore, these methods violate the implementation invariance axiom, for example because of concatenations of linear operations such as BN and Convolutional layers. However, this problem can be overcome with model canonization, i.e., re-structuring the network into a canonical form implementing exactly the same mathematical function.

Parameterization: Rule-based backpropagation approaches are highly flexible and allow tailoring the XAI method to the underlying model and the task at hand. However, this flexibility comes at the cost of numerous possible parameterizations. For instance, the \( \gamma \)-rule in LRP computes relevances \( R_j \) of layer \( j \) given relevances \( R_k \) from the succeeding layer \( k \) as

\[
R_j = \sum_k \frac{a_j \cdot (w_{jk} + \gamma w^+_{jk})}{\sum_j a_j \cdot (w_{jk} + \gamma w^+_{jk})} \cdot R_k ,
\]

where \( a_j \) are the lower-layer activations, \( w_{jk} \) are the weights between layers \( j \) and \( k \), \( w^+_{jk} \) is the positive part of \( w_{jk} \) and \( \gamma \) is a parameter allowing to regulate the impact of positive and negative contributions. Therefore, \( \gamma \) is a hyper-parameter that has to be defined for each layer. Note that the \( \gamma \)-rule becomes equivalent to the \( \alpha \) rule as \( \gamma \to \infty \), where negative contributions are disregarded. Similarly, for \( \gamma = 0 \), it is equivalent to the \( \epsilon \)-rule, where negative and positive contributions are treated equally. The choice of \( \gamma \) for each layer can highly impact various measurable aspects of explanation quality.

3. Model Canonization

We assume there is a model \( f \), which, given input data \( x \), implements the function \( f(x) \). We further assume that \( f \) contains model components which pose challenges for the implementation of certain XAI methods. Model canonization aims to replace \( f \) by a model \( g \) where \( g(x) = f(x) \), but \( g \) does not contain the problematic components. We call \( g \) the canonical form of all models implementing the function \( f(x) \). In practice, model canonization can be achieved by restructuring the model and combining several model components, as outlined in the following sections.

3.1. BatchNorm Layer Canonization

BN layers [21] were introduced to increase the stability of model training by normalizing the gradient flows in neural networks. Specifically, BN adjusts the mean and standard deviation as follows:

\[
\text{BN}(x) = w_{\text{BN}} \frac{x - \mu}{\sqrt{\sigma + \epsilon}} + b_{\text{BN}} ,
\]

where \( w_{\text{BN}} \) and \( b_{\text{BN}} \) are learnable weights and a bias term of the BN layer, \( \mu \) and \( \sigma \) are the running mean and running variance and \( \epsilon \) is a stabilizer. However, as discussed in Section 2.3, BN layers have shown to pose challenges for modified backpropagation XAI methods, such as LRP [20]. To address that problem, model canonization can be applied to remove BN layers without changing the output of the function. We make use of the fact that during test time the BN operation can be viewed as
a fixed affine transformation. Specifically, we follow previous works [15], which have shown that BN layers can be fused with neighboring linear layers, including fully connected layers and Convolutional layers, of form $w^L_L x + b_L$, where $w^L_L$ is the weight matrix and $b_L$ is the bias term. This results in a single linear layer, combining the affine transformations from the original linear layer and the BN. The exact computation of the new parameters of the linear transformation depends on the order of model components:

**Linear → BN:** Many popular architectures (including VGG [43] and ResNets [16]) apply batch normalization directly after Convolutional layers. Hence, this model component implements the following function:

$$f(x) = \text{BN} \left( \text{Linear}(x) \right)$$

$$= \left( \frac{w_{BN}}{\sqrt{\sigma + \epsilon}} w^L_L x + \frac{w_{BN}}{\sqrt{\sigma + \epsilon}} (b_L - \mu) + b_{BN} \right)$$

(3)

which can be merged into a single linear layer with weight $w_{new} = \frac{w_{BN}}{\sqrt{\sigma + \epsilon}} w^L_L$ and bias $b_{new} = \frac{w_{BN}}{\sqrt{\sigma + \epsilon}} (b_L - \mu) + b_{BN}$. See Section A in the supplementary material for details.

**BN → Linear:** Other implementations apply BN right before linear layers (i.e., after the activation function of the previous layer), which impacts the computation of parameters of the merged linear transformation:

$$f(x) = \text{Linear} \left( \text{BN}(x) \right)$$

$$= \left( \frac{w_L^T}{\sqrt{\sigma + \epsilon}} w^L_B x + \frac{w_L^T}{\sqrt{\sigma + \epsilon}} w_{BN} + b_L \right)$$

(4)

Again, this component can be fused into a single linear transformation with weight $w_{new} = \frac{w_L^T w_{BN}}{\sqrt{\sigma + \epsilon}}$ and bias $b_{new} = \frac{w_L^T w_{BN} \mu}{\sqrt{\sigma + \epsilon}} + b_L$. Note that there are practical challenges when padding is applied, for instance in a Convolutional layer. In this case, the bias becomes a spatially varying term, which cannot be implemented with standard Convolutional layers. See Section A in the supplementary material for details.

**BN → ReLU → Linear:** In some architectures, BN layers have to be merged with linear layers with an activation function (e.g., ReLU) in between. For instance, in DenseNets model components occur in that order. In that case, model canonization goes beyond merging two affine transformations, because of the non-linear activation function in between. Therefore, we propose to swap the BN layer and the activation function, which can be achieved by defining a new activation function, named $\text{ReLU}_{thresh}$ which depends on the parameters of the BN layer, such that

$$\text{ReLU}(\text{BN}(x)) = \text{BN}(\text{ReLU}_{thresh}(x))$$

(5)

where

$$\text{ReLU}_{thresh}(x) = \begin{cases} x & \text{if } (w_{BN} > 0 \text{ and } x > z) \\ x & \text{if } (w_{BN} < 0 \text{ and } x < -z) \\ z & \text{otherwise} \end{cases}$$

(7)

with $z = \frac{\mu}{w_{BN}/\sqrt{\sigma + \epsilon}}$. Hence, $\text{BN} \rightarrow \text{ReLU} \rightarrow \text{Linear}$ is first transformed into $\text{ThresholdReLU} \rightarrow \text{BN} \rightarrow \text{Linear}$; then the BN layer and the linear layer can be merged with Eq. 6.

### 3.2. Canonization of Popular Architectures

We now demonstrate the canonization of popular neural network architectures. We picked 4 image classification models (VGG, ResNet, EfficientNet and DenseNet) and one VQA model (Relation Network [34]).

**Image Classification Models:** Many popular image classification model architectures, such as VGG [43], ResNet [16] and EfficientNet [44], apply BN directly after linear layers. Therefore, these networks can easily be canonized using Eq. 4. It gets more complicated, however, if model architectures are more complex with highly interconnected building blocks. DenseNets, for example, use skip connections to pass activations from each dense block to all subsequent blocks, as shown in Fig. 1. Each block applies BN on the concatenated inputs coming from multiple blocks, followed by ReLU activation and a Convolutional layer ($\text{BN} \rightarrow \text{ReLU} \rightarrow \text{Conv}$). Note that due to the high interconnectivity, BN layers cannot easily be merged into linear layers from neighboring blocks, because most linear layers pass their activations to multiple blocks, and vice versa, most blocks receive activations from multiple blocks. Consequently, the linear transformation implementing the BN function has to be merged with the linear layer following the ReLU activation within the same block. Therefore, we propose to perform model canonization by first applying Eq. 7 and then Eq. 6 to join BN layers with following linear layers within the same block, over the ReLU function between them. This process is visualized in Fig. 1. In addition, we apply Eq. 4 to merge the first BN layer in the initial layers of the DenseNet architecture, before the dense blocks. Moreover, there is a $\text{BN} \rightarrow \text{ReLU} \rightarrow \text{AvgPool2d} \rightarrow \text{Conv}$ chain in the end of the network, which can also be merged using Eq. 7 and Eq. 6.

**VQA Model:** In contrast to image classification models, VQA models, e.g., Relation Network [34], require two paths to encode both, the input image and the input question. Relation Networks use a simple Convolutional neural network as image encoder. The implementation by the authors of CLEVR-XAI [4] applies BN after the activation function ($\text{Conv} \rightarrow \text{ReLU} \rightarrow \text{BN}$). Therefore, we merge BN layers with the Convolutional layer from the following block using Eq. 6. The BN layer of the last block of the image encoder has to be merged with the fully connected
layer of the following module, which, however, receives a concatenation of image encoding and text encoding from the input question. Therefore, it has to be assured that BN parameters are merged only with the weights operating on inputs coming from the image encoder (see Section B and Fig. A.1 in the supplementary material for details).

4. Experiments: XAI Evaluation Framework

4.1. Datasets

ILSVRC2017 [32] is a popular benchmark dataset for object recognition tasks with 1.2 million samples categorized into 1,000 classes, out of which we randomly picked 50 classes for our experiments (see Section F.1 in the supplementary material). Bounding box annotations are provided for a subset of ILSVRC2017, which we use for localization metrics. Per class, we use up to 640 random samples. Note that ILSVRC2017 faces a center-bias, i.e., most of the objects to be classified are located in the center of the image. Therefore, naive explainers can assume that models base their decisions on center pixels. To that end, we include additional experiments using the Pascal VOC 2012 dataset [13] in Section D of the supplementary material.

CLEVR-XAI [4] builds upon the CLEVR dataset [22], which is an artificial VQA dataset. It contains 10,000 images showing objects with varying characteristics regarding shape, size, color and material. Moreover, there are simple and complex questions that need to be answered. The task is framed as a classification task, in which, given an image and a question, the model has to predict the correct response out of 28 possible answers. In total, there are approx. 40,000 simple questions, asking for certain characteristics of single objects. In addition, there are 100,000 complex questions, which require the understanding of relationships between multiple objects. CLEVR-XAI further comes with ground-truth explanations, encoded as binary masks locating the objects that are required in order to answer the question. Simple questions come with two binary masks, which are GT Single Object, localizing the object affected by the question, and GT All Objects, localizing all objects in the image. For complex questions, there are four binary masks, including GT Union localizing all objects that are required to answer the question (we refer to [4] for details on the other masks).

4.2. Models

For our experiments with ILSVRC2017, we analyze VGG-16 [43], ResNet-18 [16], EfficientNet-B0 [44] and DenseNet-121 [19]. We use pre-trained models provided in the PyTorch model zoo [30]. We use a Relation Network [34] for our experiments with CLEVR-XAI.

4.3. XAI Methods and Implementation Details

We analyze rule-based and modified backpropagation based XAI methods, namely Excitation Backprop (EB) and LRP. Note that other backpropagation-based methods, such as Salieny, Smoothgrad, Integrated Gradients and Guided Backprop are not impacted by model canonization [29] and are therefore not analyzed in this experiment. For each method, we compute explanations for both, the original and the canonized model. We use zennit\[2\] as toolbox to compute explanations. For ILSVRC2017 with LRP, we analyze two pre-defined composites, i.e., mappings from layer type to LRP rule which have been established in literature [26], namely EpsilonPlus (\(\epsilon^+\)) and Alpha2-Beta1

---

\[1\]https://github.com/chr5tphr/zennit
comparing models with and without canonization using the
metrics higher scores are better.

4.5. Canonization Results

In order to convert 3-dimensional relevance scores per voxel into 2-dimensional scores per pixel (height × width), we simply sum the relevances by dividing all values by the square root of the second moment to bound their variance for numerical stability when comparing heatmaps.

4.4. XAI Metrics

In our experiments, we quantitatively measure the impact of canonization of the selected model architectures with various metrics, probing the quality of explanations from different viewpoints. We use the quantus toolbox\footnote{https://github.com/understandable-machine-intelligence-lab/quantus} to compute the following metrics: We measure Faithfulness using Region Perturbation with blurring as baseline function. We compute the Area over Perturbation Curve (AoPC)\cite{33} to measure the faithfulness in a single number as:

\[
\text{AoPC} = \frac{1}{L+1} \left( \sum_{k=0}^{L} f(x^{(0)}) - f(x^{(k)}) \right),
\]

where \(x\) is the input sample, \(k\) is the perturbation step and \(L\) is the total number of perturbations. The AoPC is averaged over all input samples. Localization quality is measured using RRA and RMA. As ground truth location, we use bounding box annotations provided for ILSVRC2017, and binary segmentation masks for CLEVR-XAI. For the latter, we use GT Unique for simple questions and GT Union for complex questions. Moreover, we use the average sensitivity to measure Robustness, sparseness for Complexity and run the logit test as a Randomization metric. While for robustness and randomization low scores are desirable, for the other metrics higher scores are better.

4.5. Canonization Results

The XAI evaluation results for the ILSVRC2017 dataset comparing models with and without canonization using the
Table 1. XAI evaluation results with and without model canonization for VGG-16, ResNet-18, EfficientNet-B0 and DenseNet-121 using the ILSVRC2017 dataset. We measure the quality of explanations using the metrics Sparseness (Complexity), Region Perturbation (Faithfulness), RRA and RMA (Localization), Avg. Sensitivity (Robustness) and Random Logit Test (Randomization). Arrows indicate whether high (↑) or low (↓) are better. Best results are shown in bold.

<table>
<thead>
<tr>
<th>Model</th>
<th>canonized</th>
<th>↑ Complexity</th>
<th>↑ Faithfulness</th>
<th>↑ Local. (RRA)</th>
<th>↑ Local. (RMA)</th>
<th>↑ Robustness</th>
<th>↓ Random.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
</tr>
<tr>
<td>VGG-16</td>
<td>EB</td>
<td>0.57 0.59</td>
<td>0.35 0.36</td>
<td>0.70 0.71</td>
<td>0.68 0.70</td>
<td>0.22 0.18</td>
<td>1.00 1.00</td>
</tr>
<tr>
<td></td>
<td>LRP-α/2/1</td>
<td>0.70 0.84</td>
<td>0.38 0.39</td>
<td>0.63 0.67</td>
<td>0.65 0.77</td>
<td>0.31 0.34</td>
<td>0.59 0.66</td>
</tr>
<tr>
<td></td>
<td>LRP-ε+</td>
<td>0.51 0.62</td>
<td>0.36 0.39</td>
<td>0.69 0.71</td>
<td>0.64 0.71</td>
<td>0.19 0.21</td>
<td>0.57 0.54</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>EB</td>
<td>0.55 0.57</td>
<td>0.29 0.29</td>
<td>0.68 0.69</td>
<td>0.66 0.67</td>
<td>0.16 0.14</td>
<td>0.97 0.97</td>
</tr>
<tr>
<td></td>
<td>LRP-α/2/1</td>
<td>0.67 0.76</td>
<td>0.32 0.32</td>
<td>0.65 0.67</td>
<td>0.69 0.75</td>
<td>0.21 0.26</td>
<td>0.65 0.61</td>
</tr>
<tr>
<td></td>
<td>LRP-ε+</td>
<td>0.51 0.58</td>
<td>0.30 0.30</td>
<td>0.69 0.70</td>
<td>0.65 0.69</td>
<td>0.14 0.15</td>
<td>0.70 0.70</td>
</tr>
<tr>
<td>EfficientNet-B0</td>
<td>EB</td>
<td>0.85 0.70</td>
<td>0.24 0.27</td>
<td>0.73 0.67</td>
<td>0.79 0.72</td>
<td>0.42 0.33</td>
<td>0.99 1.00</td>
</tr>
<tr>
<td></td>
<td>LRP-α/2/1</td>
<td>0.75 0.77</td>
<td>0.29 0.20</td>
<td>0.72 0.65</td>
<td>0.79 0.73</td>
<td>0.48 0.49</td>
<td>0.57 0.51</td>
</tr>
<tr>
<td></td>
<td>LRP-ε+</td>
<td>0.50 0.73</td>
<td>0.28 0.30</td>
<td>0.75 0.75</td>
<td>0.69 0.79</td>
<td>0.12 0.21</td>
<td>0.61 0.65</td>
</tr>
<tr>
<td>DenseNet-121</td>
<td>EB</td>
<td>0.66 0.62</td>
<td>0.15 0.31</td>
<td>0.58 0.72</td>
<td>0.53 0.73</td>
<td>0.57 0.17</td>
<td>0.75 0.89</td>
</tr>
<tr>
<td></td>
<td>LRP-α/2/1</td>
<td>0.82 0.81</td>
<td>0.25 0.33</td>
<td>0.64 0.71</td>
<td>0.68 0.81</td>
<td>0.65 0.28</td>
<td>0.40 0.44</td>
</tr>
<tr>
<td></td>
<td>LRP-ε+</td>
<td>0.67 0.66</td>
<td>0.26 0.33</td>
<td>0.70 0.74</td>
<td>0.71 0.77</td>
<td>0.63 0.19</td>
<td>0.39 0.48</td>
</tr>
</tbody>
</table>

Table 2. XAI evaluation results for Relation Network with and without model canonization using the CLEVR-XAI dataset for simple and complex questions using pos-l2-norm-sq-pooling. Arrows indicate whether high (↑) or low (↓) are better. Best results are shown in bold.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
<td>no yes</td>
</tr>
<tr>
<td>Simple</td>
<td>EB</td>
<td>0.99 0.97</td>
<td>0.50 0.51</td>
<td>0.64 0.61</td>
<td>0.76 0.70</td>
<td>1.37 1.39</td>
<td>1.00 1.00</td>
</tr>
<tr>
<td></td>
<td>LRP-Custom [4]</td>
<td>0.95 0.98</td>
<td>0.52 0.52</td>
<td>0.70 0.70</td>
<td>0.75 0.83</td>
<td>1.33 1.35</td>
<td>0.99 1.00</td>
</tr>
<tr>
<td>Complex</td>
<td>EB</td>
<td>0.99 0.97</td>
<td>0.44 0.45</td>
<td>0.66 0.62</td>
<td>0.82 0.77</td>
<td>1.36 1.35</td>
<td>1.00 0.99</td>
</tr>
<tr>
<td></td>
<td>LRP-Custom [4]</td>
<td>0.94 0.97</td>
<td>0.45 0.46</td>
<td>0.54 0.63</td>
<td>0.79 0.86</td>
<td>1.33 1.34</td>
<td>0.98 0.99</td>
</tr>
</tbody>
</table>

(Conv 8-10) layers, as well as fully-connected layers in the classification head. We define one γ-parameter per group with \(\gamma \in \{0, 0.1, 0.25, 0.5, 1, 10\}\), and run a grid search for all possible combinations with and without model canonization, i.e., \(2 \times 6^4 = 2592\) \(\gamma\)-configurations. Note that in theory, we could also evaluate different sub-canonizations, where we only canonize certain parts of the model. This, however, further increases the degrees of freedom. Further, note that more advanced multi-metric-objective hyperparameter optimization approaches can be employed. However, we decided to go forward with simple grid search, because our goal is to highlight the importance of the choice of XAI hyperparameters and the impact on various evaluation metrics. We evaluate the resulting explanations with the metrics described in section 4.4 and show the results in Fig. 2. Specifically, each line represents the score per metric with \(\gamma\) for a certain group of layers kept constant, averaged over all \(\gamma\)-parameterizations for other layer groups. It can be seen that the impact of the choice of \(\gamma\) depends on the position of the layer in the network and, in addition, differs by the metric of choice. For instance, the robustness of the explanations is mainly impacted by the \(\gamma\)-value in low-level layers (Conv 1-3), while it has no impact for the other layers. In contrast, randomization is mostly impacted by the choice of \(\gamma\) for fully connected layers in the classification head. Interestingly, canonization has a large impact on the optimal choice of \(\gamma\) for low-level layers when measuring the faithfulness of the resulting explanations. In Fig. 3 we show attribution heatmaps for three samples using different \(\gamma\)-configurations, employing the best and worst parameterization according to the metrics faithfulness, localization, and complexity. Each metric favors another parametrization, leading to different attribution heatmaps. High \(\gamma\)-values in low-level layers (\(\gamma_1\)) appear to be favorable for all metrics, i.e., more focus on positive contributions on those layers. This leads to attribution heatmaps with less noise, which is beneficial w.r.t faithfulness, localization, and complexity.

5. Conclusions

In this work, we proposed an evaluation framework for XAI methods which can be leveraged to optimize the quality of explanations based on a variety of XAI metrics. Specifically, we demonstrated the application of our framework to measure the impact of model canonization towards various aspects of explanation quality. Therefore, we extended the model canonization approach to state-
of-the-art model architectures, including EfficientNets and DenseNets. Despite not always being beneficial w.r.t. all examined architectures, model canonization provides an extra option when adopting XAI methods to the task at hand. Moreover, we applied our evaluation framework for hyperparameter optimization for XAI methods and demonstrated the impact of parameters w.r.t different XAI metrics. While we have evaluated our methods for LRP, it is also applicable to other configurable XAI methods, such as DeepLift. Future work will focus on the canonization of additional relevant model architectures, e.g., Vision Transformer [12]. In addition, optimizing the hyperparameter search is a promising research direction, e.g., with random search, evolutionary algorithms, or other approaches to reduce the search space. Moreover, the framework can be applied with other optimization objectives, e.g., to find LRP configurations that mimic other, more expensive XAI methods, e.g., SHAP.

Acknowledgements

This work was supported by the European Union’s Horizon 2020 research and innovation programme (EU Horizon 2020) as grant [iToBoS (965221)]; the state of Berlin within the innovation support program ProFIT (IBB) as grant [BerDiBa (10174498)]; the Federal Ministry of Education and Research (BMBF) as grant [BIFOLD (01IS18025A, 01IS18037I)]; the German Research Foundation (ref. DFG KI-FOR 5363); the Research Council of Norway, via the SFI Visual Intelligence grant [project grant number 309439].
References


