Explainable AI Methods - A Brief Overview

Andreas Holzinger[1,2,3][0000−0002−6786−5194], Anna Saranti[1,2],[0000−0002−1085−8428], Christoph Molnar[4][0000−0003−2331−868X], Przemyslaw Biecek[5,6][0000−0001−8423−1823], Wojciech Samek[7,8][0000−0002−6283−3265]

[1] Human-Centered AI Lab, University of Natural Resources and Life Sciences, Vienna, Austria
[2] Medical University Graz, Austria
[3] xAI Lab, Alberta Machine Intelligence Institute, Edmonton, Canada
[4] Leibniz Institute for Prevention Research and Epidemiology – BIPS GmbH, Bremen, Germany
[5] Warsaw University of Technology, Poland
[6] University of Warsaw, Poland
[7] Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute, Berlin, Germany
[8] BIFOLD – Berlin Institute for the Foundations of Data and Learning, Germany

Abstract. Explainable Artificial Intelligence (xAI) is an established field with a vibrant community that has developed a variety of very successful approaches to explain and interpret predictions of complex machine learning models such as deep neural networks. In this article, we briefly introduce a few selected methods and discuss them in a short, clear and concise way. The goal of this article is to give beginners, especially application engineers and data scientists, a quick overview of the state of the art in this current topic. The following 17 methods are covered in this chapter: LIME, Anchors, GraphLIME, LRP, DTD, PDA, TCAV, XGNN, SHAP, ASV, Break-Down, Shapley Flow, Textual Explanations of Visual Models, Integrated Gradients, Causal Models, Meaningful Perturbations, and X-NeSyL.

Keywords: Explainable AI, Methods, Evaluation.

1 Introduction

Artificial intelligence (AI) has a long tradition in computer science. Machine learning (ML) and particularly the success of "deep learning" in the last decade made AI extremely popular again [15], [25], [90].

The great success came with additional costs and responsibilities: the most successful methods are so complex that it is difficult for a human to re-trace, to understand, and to interpret how a certain result was achieved. Consequently, explainability/interpretability/understandability is motivated by the lack of transparency of these black-box approaches, which do not foster trust and acceptance of AI in general and ML in particular. Increasing legal and data protection

aspects, e.g., due to the new European General Data Protection Regulation (GDPR, in force since May 2018), complicate the use of black-box approaches, particularly in domains that affect human life, such as the medical field [56], [63], [73], [76].

The term explainable AI (xAI) was coined by DARPA [28] and gained meanwhile a lot of popularity. However, xAI is not a new buzzword. It can be seen as a new name for a very old quest in science to help to provide answers to questions of why [66]. The goal is to enable human experts to understand the underlying explanatory factors of why an AI decision has been made [64]. This is highly relevant for causal understanding and thus enabling ethical responsible AI and transparent verifiable machine learning in decision support [74].

The international community has developed a very broad range of different methods and approaches and here we provide a short concise overview to help engineers but also students to select the best possible method. Figure 1 shows the most popular XAI toolboxes.

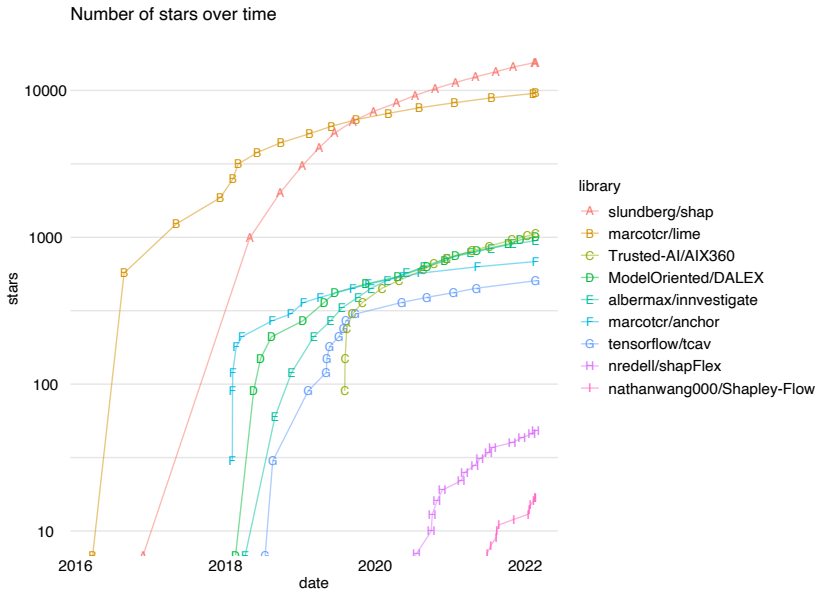

Fig. 1. Number of stars on GitHub for the most popular repositories presented in this paper. While these repositories focus on the explanation task, the new Quantus toolbox [30] offers a collection of methods for evaluating and comparing explanations.

In the following we provide a short overview of some of the most popular methods for explaining complex models. We hope that this list will help both practitioners in choosing the right method for model explanation and XAI method developers in noting the shortcomings of currently available methods.

Figure 2 gives an overview of the chronology of development of successive explanatory methods. Methods such as LRP and LIME were among the first[9] generic techniques to explain decisions of complex ML models. In addition to the overview of explanation techniques, we would also like to hint the interested reader at work that developed methods and offered datasets to objectively evaluate and systematically compare explanations. To mention here is Quantus[10] [30], a new toolbox offering an exhaustive collection of evaluation methods and metrics for explanations, and CLEVR-XAI[11] [8], a benchmark dataset for the ground truth evaluation of neural network explanations.

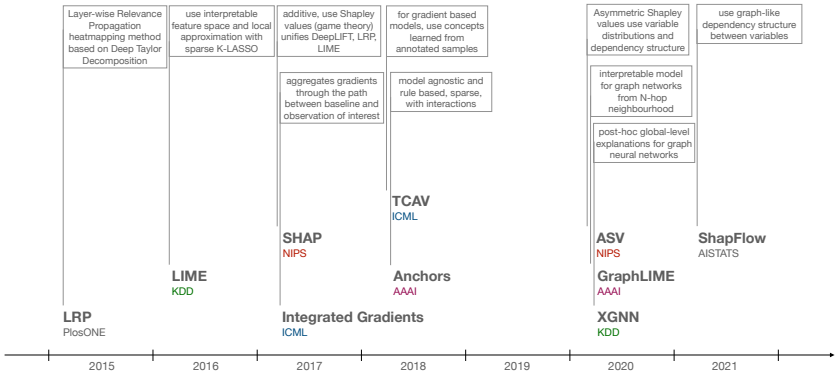

Fig. 2. Chronology of the development of successive explanatory methods described in this paper. Initially, the methods were focused on model analysis based on the model itself or on sample data. Subsequent methods used more and more information about the structure and relationships between the analysed variables.

2 Explainable AI Methods - overview

2.1 LIME (Local Interpretable Model Agnostic Explanations)

Idea: By treating the machine learning models as black-box functions, model agnostic explanation methods typically only have access to the model's output.

[9] We are aware that gradient-based sensitivity analysis and occlusion-based techniques have been proposed even earlier [62, 11, 75, 89]. However, theses techniques have various disadvantages (see [61, 70]) and are therefore not considered in this paper.

[10] `https://github.com/understandable-machine-intelligence-lab/quantus`

[11] `https://github.com/ahmedmagdiosman/clevr-xai`

The fact that these methods do not require any information about the model's internals, e.g., in the case of neural networks the topology, learned parameters (weights, biases) and activation values, makes them widely applicable and very flexible.

One prominent representative of this class of explanation techniques is the Local Interpretable Model-agnostic Explanations (LIME) method [67]. The main idea of LIME is to explain a prediction of a complex model f_M, e.g., a deep neural network, by fitting a local surrogate model f_S, whose predictions are easy to explain. Therefore, LIME is also often referred to as surrogate-based explanation technique [70]. Technically, LIME generates samples in the neighborhood $\mathcal{N}_{\mathbf{x}_i}$ of the input of interest $\mathbf{x}_i$, evaluates them using the target model, and subsequently approximates the target model in this local vicinity by a simple linear function, i.e., a surrogate model which is easy to interpret. Thus, LIME does not directly explain the prediction of the target model $f_M(\mathbf{x}_i)$, but rather the predictions of a surrogate model $f_S(\mathbf{x}_i)$, which locally approximates the target model (i.e., $f_M(\mathbf{x}) \approx f_S(\mathbf{x})$ for $\mathbf{x} \in \mathcal{N}_{\mathbf{x}_i}$).

GitHub Repo: https://github.com/marcotcr

Discussion: There are meanwhile many successful applications of LIME in different application domains which demonstrates the popularity of this model agnostic method. As a limitation can be seen that LIME only indirectly solves the explanation problem by relying on a surrogate model. Thus, the quality of the explanation largely depends on the quality of the surrogate fit, which itself may require dense sampling and thus may result in large computational costs. Furthermore, sampling always introduces uncertainty, which can lead to non-deterministic behaviours and result in variable explanations for the same input sample.

2.2 Anchors

Idea: The basic idea is that individual predictions of any black-box classification model are explained by finding a decision rule that sufficiently "anchors" the prediction - hence the name "anchors" [68]. The resulting explanations are decision rules in the form of IF-THEN statements, which define regions in the feature space. In these regions, the predictions are fixed (or "anchored") to the class of the data point to be explained. Consequently, the classification remains the same no matter how much the other feature values of the data point that are not part of the anchor are changed.

Good anchors should have high precision and high coverage. *Precision* is the proportion of data points in the region defined by the anchor that have the same class as the data point being explained. *Coverage* describes how many data points an anchor's decision rule applies to. The more data points an anchor covers, the better, because the anchor then covers a larger area of the feature space and thus represents a more general rule. Anchors is a model-agnostic explanation method,

i.e., it can be applied to any prediction model without requiring knowledge about the internals. Search and construction of decision rules is done by reinforcement learning (RL) [81], [32] in combination with a modified beam search, a heuristic search algorithm that extends the most promising nodes in a graph. The anchors algorithm cycles through different steps: produce candidate anchors, select the best candidates, then use beam search to extend the anchor rules. To select the best candidate, it is necessary to call the model many times, which can be seen as an exploration or multi-armed bandit problem.

GitHub Repo: `https://github.com/marcotcr/anchor`

Discussion: The anchors are model-independent and can be applied to different domains such as tabular data, images and text, depending on the perturbation strategy. However, in the current Python implementation, anchors only supports tabular and text data. Compared to LIME, the scope of interpretation is clearer as the anchors specify the boundaries within which they should be interpreted. The coverage of an anchor decision rule can be used as a measure of the model fidelity of the anchor. Furthermore, the decision rules are easy to understand, but there are many hyper-parameters in the calculation of anchors, such as the width of the beam and the precision threshold, which need to be tuned individually. The perturbation strategies also need to be carefully selected depending on the application and model. The calculation of anchors requires many calls to the prediction function, which makes the anchors computationally intensive. Data instances that are close to the decision boundary of the model may require more complex rules with more features and less coverage. Unbalanced classification problems can produce trivial decision rules, such as classifying each data point as the majority class. A possible remedy is to adapt the perturbation strategy to a more balanced distribution.

2.3 GraphLIME

Idea: GraphLIME [38] is a method that takes the basic idea of LIME (see Section 2.1) but is not linear. It is applied to a special type of neural network architecture, namely graph neural networks (GNN). These models can process non-Euclidean data as they are organised in a graph structure [9]. The main tasks that GNNs perform are node classification, link prediction and graph classification. Like LIME, this method tries to find an interpretable model, which in this case is the Hilbert-Schmidt Independence Criterion (HSIC) Lasso model, for explaining a particular node in the input graph. It takes into account the fact that during the training of the GNN, several nonlinear aggregation and combination methods use the features of neighbouring nodes to determine the representative embedding of each node. This embedding is used to distinguish nodes into different classes in the case of node classification and to collectively distinguish graphs in graph classification tasks.

Since for this type of model a linear explanation as LIME would return unfaithful results, the main idea of GraphLIME is to sample from the N-hop neighbourhood of the node and collect features w.r.t. to the node prediction. Those are used to train the HSIC Lasso model, which is a kernel method - thereby interpretable - that can compute on which node features the output prediction depends on. This is similar to the perturbation method that LIME uses, while the comparison, in this case, is based on HSIC estimation between the random variables representing the features and the prediction distributions. This method learns correlations between the features of the neighbours which also underline its explanation capabilities.

GitHub Repo: `https://github.com/WilliamCCHuang/GraphLIME`

Discussion: The developers compared GraphLIME with one of the first xAI methods for GNNs at the time, namely GNNExplainer, w.r.t. three criteria: (1) ability to detect useless features, (2) ability to decide whether the prediction is trustworthy, and (3) ability to identify the better model among two GNN classifiers. They show that for synthetic data and human-labelled annotations, GraphLIME exceeds the GNNExplainer by far in the last two criteria. They arrive at a very interesting insight, namely that models that have fewer untrustworthy features in their explanation have better classification performance. Furthermore, GraphLIME is shown to be computationally much more efficient than GNNExplainer. It would be beneficial - and is considered future work - if GraphLIME was also trying to find important graph substructures instead of just features, if it was compared with other methods like PGExplainer [52], PGMExplainer [83], GNN-LRP [72], and if it were extended to multiple instance explanations. Finally, it is important to note that GraphLIME is successfully used for the investigation of backdoor attacks on GNNs by uncovering the relevant features of the graph's nodes [86].

2.4 Method: LRP (Layer-wise Relevance Propagation)

Idea: Layer-wise Relevance Propagation (LRP) [10] is a propagation-based explanation method, i.e., it requires access to the model's internals (topology, weights, activations etc.). This additional information about the model, however, allows LRP to simplify and thus more efficiently solve the explanation problem. More precisely, LRP does not explain the prediction of a deep neural network in one step (as model agnostic methods would do), but exploits the network structure and redistributes the explanatory factors (called relevance R) layer by layer, starting from the model's output, onto the input variables (e.g., pixels). Each redistribution can be seen as the solution of a simple (because only between two adjacent layers) explanation problem (see interpretation of LRP as Deep Taylor Decomposition in Section 2.5).

Thus, the main idea of LRP is to explain by decomposition, i.e., to iteratively redistribute the total evidence of the prediction $f(\mathbf{x})$, e.g., indicating that there

is a cat in the image, in a conservative manner from the upper to the next lower layer, i.e.,

$$\sum_i R_i^{(0)} = \ldots = \sum_j R_j^{(l)} = \sum_k R_k^{(l+1)} = \ldots = f(\mathbf{x}). \tag{1}$$

Note that $R_i^{(0)}$ denotes the relevance assigned to the ith input element (e.g., pixel), while $R_j^{(l)}$ stands for relevance assigned to the jth neuron at the lth layer. This conservative redistribution not only ensures that no relevance is added or lost on the way (analogous to energy conservation principle or Kirchhoff's law in physics), but also allows for signed explanations, where positive relevance values hint at relevant information supporting the prediction and negative relevance values indicate evidence speaking against it. Different redistribution rules, adapted to the specific properties of particular neural network layers, have been proposed for LRP [58, 42]. In contrast to other XAI techniques which are purely based on heuristics, the LRP rules have a clear theoretical foundation, namely they result from the Deep Taylor Decomposition (DTD) [60] of the relevance function with a particular choice of root point (see Section 2.5).

While LRP has been originally developed for convolutional neural networks and bag-of-words type of models, various extensions have been proposed, making it a widely applicable XAI techniqe today. For instance, Arras et al. [7, 6] developed meaningful LRP redistribution rules for LSTM models. Also LRP variants for GNN and Transformer models have been recently proposed [72, 3]. Finally, through the "neuralization trick", i.e., by converting a non-neural network model into a neural network, various other classical ML algorithms have been made explainable with LRP, including k-means clustering [39], one-class SVM [40] as well as kernel density estimation [59]. Furthermore, meta analysis methods such as spectral relevance analysis (SpRAy) [47] have been proposed to cluster and systematically analyze sets of explanations computed with LRP (SpRAY is not restricted to LRP explanations though). These analyses have been shown useful to detect artifacts in the dataset and uncover so-called "Clever Hans" behaviours of the model [47].

The recently published Zennit toolbox [4] implements LRP (and other methods) in Python, while the CoRelAy[12] toolboxi offers a collection of meta analysis methods. Furthermore, the GitHub library iNNvestigate provides a common interface and out-of-the-box implementation for many analysis methods, including LRP [2].

GitHub Repo: `https://github.com/chr5tphr/zennit`
`https://github.com/albermax/innvestigate`

Discussion: LRP is a very popular explanation method, which has been applied in a broad range of domains, e.g., computer vision [46], natural language processing [7], EEG analysis [78], meteorology [54], among others.

[12] `https://github.com/virelay/corelay`

The main advantages of LRP are its high computational efficiency (in the order of one backward pass), its theoretical underpinning making it a trustworthy and robust explanation method (see systematic comparison of different XAI methods [8]), and its long tradition and high popularity (it is one of the first XAI techniques, different highly efficient implementations are available, and it has been successfully applied to various problems and domains). The price to pay for the advantages is a restricted flexibility, i.e., a careful adaptation of the used redistribution rules may be required for novel model architectures. For many popular layers types recommended redistribution rules are described in [58, 42].

Finally, various works showed that LRP explanations can be used beyond sheer visualization purposes. For instance, [47] used them to semi-automatically discover artefacts in large image corpora, while [79, 5] went one step further and demonstrated that they can be directly (by augmenting the loss) or indirectly (by adapting training data) used to improve the model. Another line of work [87, 14] exploits the fact that LRP computes relevance values not only for the input variables, but for all elements of the neural network, including weights, biases and individual neurons, to optimally prune and quantize the neural model. The idea is simple, since LRP explanations tell us which parts of the neural network are relevant, we can simply remove the irrelevant elements and thus improve the coding efficiency and speed up the computation.

2.5 Deep Taylor Decomposition (DTD)

Idea: The Deep Taylor Decomposition (DTD) method [60] is a propagation-based explanation technique, which explains decisions of a neural network by decomposition. It redistributes the function value (i.e., the output of the neural network) to the input variables in a layer-by-layer fashion, while utilizing the mathematical tool of (first-order) Taylor expansion to determine the proportion or relevance assigned to the lower layer elements in the redistribution process (i.e., their respective contributions). This approach is closely connected to the LRP method (see Section 2.4). Since most LRP rules can be interpreted as a Taylor decomposition of the relevance function with a specific choice of root point, DTD can be seen as the mathematical framework of LRP.

DTD models the relevance of a neuron k at layer l as a simple relevance function of the lower-layer activations, i.e.,

$$R_k(\mathbf{a}) = \max(0, \sum_i a_i w_{ik}) c_k, \tag{2}$$

where $\mathbf{a} = [a_1 \dots a_d]$ are the activations at layer $l-1$, w_{ik} are the weights connecting neurons i (at layer $l-1$) and k (at layer l), and c_k is a constant. This model is certainly valid at the output layer (as R_k is initialized with the network output $f(\mathbf{x})$). Through an inductive argument the authors of [60] proved that this model also (approximatively) holds at intermediate layers. By representing

this simple function as Taylor expansion around a root point $\tilde{\mathbf{a}}$, i.e.,

$$R_k(\mathbf{a}) = \underbrace{R_k(\tilde{\mathbf{a}})}_{0} + \underbrace{\sum_i (a_i - \tilde{a}_i) \cdot \nabla [R_k(\mathbf{a})]_i}_{\text{redistributed relevance}} + \underbrace{\epsilon}_{0}, \tag{3}$$

DTD tells us how to meaningfully redistribute relevance from layer l to layer $l-1$. This redistribution process is iterated until the input layer. Different choices of root point are recommended for different types of layers (conv layer, fully connected layer, input layer) and lead to different LRP redistribution rules [58].

GitHub Repo: `https://github.com/chr5tphr/zennit`
`https://github.com/albermax/innvestigate`

Discussion: DTD is a theoretically motivated explanation framework, which redistributes relevance from layer to layer in a meaningful manner by utilizing the concept of Taylor expansion. The method is highly efficient in terms of computation and can be adapted to the specific properties of a model and its layers (e.g., by the choice of root point). As for LRP, it is usually not straight forward to adapt DTD to novel model architectures (see e.g. local renormalization layers [18]).

2.6 Prediction Difference Analysis (PDA)

Idea: At the 2017 ICLR conference, Zintgraf et al. [91] presented the Prediction Difference Analysis (PDA) method. The method is based on the previous idea presented by [69] where, for a given prediction, each input feature is assigned a relevance value with respect to a class c. The idea of PDA is that the relevance of a feature x_i can be estimated by simply measuring how the prediction changes when the feature is unknown, i.e., the difference between $p(c|\mathbf{x})$ and $p(c|\mathbf{x}_{\setminus i})$, where $\mathbf{x}_{\setminus i}$ denotes the set of all input features except x_i. Now to evaluate the prediction, specifically to find $p(c|\mathbf{x}_{\setminus i})$ there are three possibilities: (1) label the feature as unknown, (2) re-train the classifier omitting the feature, or (3) simulate the absence of a feature by marginalizing the feature. With that a relevance vector $(\mathrm{WE}_i)_{i=1\ldots m}$ (whereby m represent the number of features) is generated, that is of the same size as the input and thus reflects the relative importance of all features. A large prediction difference indicates that the feature contributed significantly to the classification, while a small difference indicates that the feature was not as important to the decision. So specifically, a positive value WE_i means that the feature contributed to the evidence for the class of interest and much more so that removing the feature would reduce the classifier's confidence in the given class. A negative value, on the other hand, means that the feature provides evidence against the class: Removing the feature also removes potentially contradictory or disturbing information, and makes the classifier more confident in the class under study.

GitHub Repo: https://github.com/lmzintgraf/DeepVis-PredDiff

Discussion: Making neural network decisions interpretable through visualization is important both to improve models and to accelerate the adoption of black-box classifiers in application areas such as medicine. In the original paper the authors illustrate the method in experiments on natural images (ImageNet data), as well as medical images (MRI brain scans). A good discussion can be found in: https://openreview.net/forum?id=BJ5UeU9xx

2.7 TCAV (Testing with Concept Activation Vectors)

Idea: TCAV [41] is a concept-based neural network approach that aims to quantify how strongly a concept, such as colour, influences classification. TCAV is based on the idea of concept activation vectors (CAV), which describe how neural activations influence the presence or absence of a user-specific concept. To calculate such a CAV, two data sets must first be collected and combined: One dataset containing images representing the concept and one dataset consisting of images in which this concept is not present. Then a logistic regression model is trained on the combined dataset to classify whether the concept is present in an image. The activations of the user-defined layer of the neural network serve as features for the classification model. The coefficients of the logistic regression model are then the CAVs. For example, to investigate how much the concept "stripped" contributes to the classification of an image as "zebra" by a convolutional neural network, a dataset representing the concept "stripped" and a random dataset in which the concept "stripped" is not present must be assembled. From the CAVs, the conceptual sensitivity can be calculated, which is the product of the CAV and the derivative of the classification (of the original network) with respect to the specified neural network layer and class. Conceptual sensitivity thus indicates how strongly the presence of a concept contributes to the desired class.

While the CAV is a local explanation as it relates to a single classification, the TCAV combines the CAVs across the data into a global explanation method and thus answers the question of how much a concept contributed overall to a given classification. First, the CAVs are calculated for the entire dataset for the selected class, concept and level. Then TCAV calculates the ratio of images with positive conceptual sensitivity, which indicates for how many images the concept contributed to the class. This ratio is calculated multiple times, each time using a different "negative" sample where the concept is not present, and a two-tailed Student t-test [77] is applied to test whether the conceptual sensitivity is significantly different from zero (the test part is where the "T" in TCAV comes from).

GitHub Repo: https://github.com/tensorflow/tcav

Discussion: TCAV can be applied to detect concept sensitivity for image classifiers that are gradient-based, such as deep neural networks. TCAV can also be used to analyze fairness aspects, e.g. whether gender or attributes of protected groups are used for classification. Very positive is that TCAV can be used by users without machine learning expertise, as the most important part is collecting the concept images, where domain expertise is important. TCAV allows to test a classification model for arbitrary concepts, even if the model was not explicitly trained on them. The technique can be used to study whether a network learned "flawed" concepts, such as spurious correlations. Detecting flawed concepts can help to improve the model. For example, it could be studied how important the presence of snow was for classifying wolves on images, and if it turns out to be important, adding images with wolves without snow might improve the robustness of the model. One drawback can be seen in the effort for labeling and collecting new data. Some concepts might also be too abstract to test, as the collection of a concept dataset might be difficult. How would one, for example, collect a dataset of images representing the concept"happiness"? Furthermore, TCAV may not work well with shallower neural networks, as only deeper networks learn more abstract concepts. Also, the technique is also not applicable to text and tabular data, but mainly to image data[13] (last accessed: 21-Feb-2022). A practical example from the medical domain can be found in [19].

2.8 XGNN (Explainable Graph Neural Networks)

Idea: The XGNN method [88] is a post-hoc method that operates on the model level, meaning that it does not strive to provide individual example-level explanations. RL drives a search to find an adequate graph starting by a randomly chosen node or a relatively small graph, as defined by prior knowledge. The RL algorithm follows two rewards at the same time: first, it tries to increase the performance of the GNN, but secondly to keep generating valid graphs, depending on the domain requirements. The action space contains only edge addition for edges in the existing graph or an enhancement with a new node. In the case where the action has a non-desirable contribution, a negative reward is provided.

GitHub Repo: `https://github.com/divelab/DIG/tree/dig/benchmarks/xgraph/supp/XGNN` and pseudocode in the paper.

Discussion: This explanation method is invented particularly for the task of graph classifications. The returned graphs are the ones that were the most representative for the GNN decision and usually have a particular property that is ingrained to make the validation possible. It is worth to mention that this is the only method that provides mode-level explanations for GNN architectures. The use of RL is justified by the fact that the search for the explanation graph is

[13] For discussion see: https://openreview.net/forum?id=S1viikbCW

non-differentiable, since it is not only driven by the performance but also by the plausibility and validity of the generated graph. Because the training of GNNs involves aggregations and combinations, this is an efficient way to overcome the obstacle of non-differentiation. The provided explanation is considered to be more effective for big datasets, where humans don't have the time to check each example's explanation individually. A disadvantage can be seen by the fact that the research idea is based on the assumption that network motifs that are the result of this explanation method are the ones on which the GNN is most "responsive"; nevertheless, this is not entirely true, since one does not know if other graph information was also important for the decision of the network. The results of the explanations are also non-concrete since in many cases ground truth is missing. That leads to a rather weak validation that bases on abstract concepts and properties of the discovered graphs, such as if they contain cycles or not.

2.9 SHAP (Shapley Values)

Note that the concepts described in this section also apply to the methods presented in Sections 2.9-2.12.

Methods in this family are concerned with explanations for the model f at some individual point x^*. They are based on a value function e_S where S is a subset of variable indexes $S \subseteq \{1, ..., p\}$. Typically, this function is defined as the expected value for a conditional distribution in which conditioning applies to all variables in a subset of S

$$e_S = E[f(x)|x_S = x_S^*]. \tag{4}$$

Expected value is typically used for tabular data. In contrast, for other data modalities, this function is also often defined as the model prediction at x^* after zeroing out the values of variables with indices outside S. Whichever definition is used, the value of e_S can be thought of as the model's response once the variables in the subset S are specified.

The purpose of attribution is to decompose the difference $f(x*) - e_\varnothing$ into parts that can be attributed to individual variables (see Figure 3A).

Idea: Assessing the importance of variable i is based on analysing how adding variable i to the set S will affect the value of the function e_S. The contribution of a variable i is denoted by $\phi(i)$ and calculated as weighted average over all possible subsets S

$$\phi(i) = \sum_{S \subseteq \{1,...,p\}/\{i\}} \frac{|S|!(p-1-|S|)!}{p!} \left(e_{S\cup\{i\}} - e_S\right). \tag{5}$$

This formula is equivalent to

$$\phi(i) = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} e_{\text{before}(\pi,i)\cup\{i\}} - e_{\text{before}(\pi,i)}, \tag{6}$$

where Π is a set of all orderings of p variables and before(π, i) stands for subset of variables that are before variable i in the ordering π. Each ordering corresponds to set of values e_S that shift from $e_\varnothing$ to $f(x^*)$ (see Figure 3B).

In summary, the analysis of a single ordering shows how adding consecutive variables changes the value of the e_S function as presented in Figure 3B. SHAP [51] arises as an averaging of these contributions over all possible orderings. This algorithm is an adaptation of Shapley values to explain individual predictions of machine learning models. Shapley values were initially proposed to distribute payouts fairly in cooperative games and are the only solution based on axioms of efficiency, symmetry, dummy, and additivity.

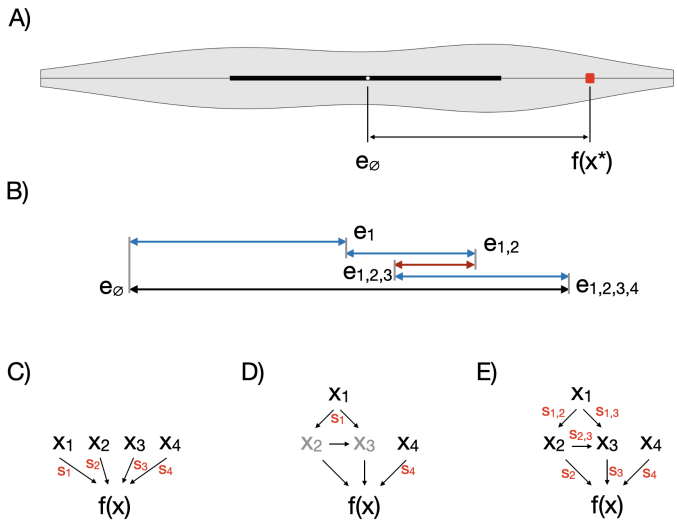

Fig. 3. Panel A. The methods presented in sections 2.9 - 2.12 explain the difference in prediction between a particular observation (x) and a baseline value. Often for the baseline value is taken the expected value from the model's prediction distribution. The methods described here distribute this difference $e_\varnothing - f(x^*)$ among the variables in the model. Panel B. Attributions are based on the changes in the expected value of the model prediction due to successive conditioning. For a given sequence of variable order (here, 1, 2, 3, 4) one can calculate how adding another variable will change the expected prediction of the model. Panel C. For the SHAP method, the variables have no structure, so any sequence of variables is treated as equally likely. Panel D. The ASV method takes into account a causal graph for variables. Only variable orderings that are consistent with this dependency graph are considered in the calculation of attributions. Causal graph controls where to assign attributions in the case of dependent variables. Panel E. The Shapley Flow method also considers a causal graph. It allocates attributions to the edges in this graph, showing how these attributions propagate through the graph.

GitHub Repo: `https://github.com/slundberg/shap`

Discussion: SHAP values sum up to the model prediction, i.e.

$$f(x^*) = e_\varnothing + \sum_i \phi(i). \tag{7}$$

In some situations, this is a very desirable property, e.g. if a pricing model predicts the value of a certain product, it is desirable to decompose this prediction additively into components attributable to individual variables. SHAP draws from a rich theoretical underpinning in game theory and fulfils desirable axioms, for example, that features that did not contribute to the prediction get an attribution of zero. Shapley values can be further combined to global interpretations of the model, such as feature dependence plots, feature importance and interaction analysis.

One large drawback of Shapley values is their immense computational complexity. For modern models such as deep neural networks and high dimensional inputs, the exact computation of Shapley values is intractable. However, model-specific implementations exist for tree-based methods (random forest, xgboost etc.) or additive models [49]. With care, one should use certain estimation versions of SHAP, such as KernelSHAP, because those are slow to compute. Furthermore, when features are dependent, Shapley values will cause extrapolation to areas with low data density. Conditional versions exist [50] (for tree-based models only), but the interpretation changes which is a common pitfall [57]. SHAP explanations are not sparse since to each feature that changes the prediction, a Shapley value different from zero is attributed, no matter how small the influence. If sparse explanations are required, counterfactual explanations might be preferable.

2.10 Asymmetric Shapley values (ASV)

Idea: SHAP values are symmetrical. This means that if two variables have the same effect on the model's behaviour, e.g. because they take identical values, they will receive equal attributions. However, this is not always a desirable property. For example, if we knew that one of the variables has a causal effect on the other, then it would make more sense to assign the entire attribution to the source variable.

Asymmetric Shapley values (ASV) [22], [23] allow the use of additional knowledge about the causal relations between variables in the model explanation process. A cause-effect relationship described in the form of causal graph allows the attribution of variables to be redistributed in such a way that the source variables have a greater attribution, providing effect on both the other dependent variables and the model predictions (see Figure 3D). SHAP values are a special case of ASV values, where the casual graph is reduced to a set of unrelated vertices (see Figure 3C).

The ASV values for variable i are also calculated as the average effect of adding a variable to a coalition of other variables, in the same way as expressed in Equation (6). The main difference is that not all possible orders of variables are considered, but only the orders are consistent with the casual graph. Thus, a larger effect will be attributed to the source variables.

GitHub Repo: `https://github.com/nredell/shapFlex`

Discussion: In order to use the ASV, a causal graph for the variables is needed. Such a graph is usually created based on domain knowledge. Examples include applications in bioinformatics with signalling pathways data for which the underlying causal structure is experimentally verified or application in social sciences, where sociodemographic data in which the direction of the relationship can be determined based on expert knowledge (e.g., age affects income rather than income affects age).

A particular application of the ASV value is the model fairness analysis. If a protected attribute, such as age or sex, does not directly affect the model's score, its SHAP attribute will be zero. But if the protected attribute is the cause for other proxy variables, then the ASV values will capture this indirect effect on the model.

2.11 Break-Down

Idea: Variable contribution analysis is based on examining the change in e_S values along with a growing set of variables described by a specific order (see Figure 3B). If the model f has interactions, different orderings of the variables may lead to different contributions. The SHAP values average over all possible orderings (see Equation 6), thus leads to additive contributions and neglecting the interactions.

An alternative is to analyze different orderings to detect when one variable has different contributions depending on what other variables precede it. This is a sign of interaction. The Break-Down method (see [16, 17]) analyzes the various orders to identify and visualize interactions in the model. The final attributions are determined based on a single ordering which is chosen based on greedy heuristics.

GitHub Repo: `https://github.com/ModelOriented/DALEX`

Discussion: Techniques such as SHAP generate explanations in the form of additive contributions. However, these techniques are often used in the analysis of complex models, which are often not additive. [26] shows that for many tabular datasets, an additive explanation may be an oversimplification, and it may lead to a false belief that the model behaves in an additive way.

2.12 Shapley Flow

Idea: Like for Asymmetric Shapley Values (ASV), Shapley Flow [84] also allows the use of the dependency structure between variables in the explanation process. As in ASV, the relationship is described by a causal graph. However, unlike ASV and other methods, attribution is assigned not to the nodes (variables) but the edges (relationships between variables). An edge in a graph is significant if its removal would change the predictions of the model (see Figure 3E). The edge attribution has the additional property that for each explanation, boundaries hold the classical Shapley values. The most extreme explanation boundary corresponds to the ASV method. The Shapley Flow method determines the attributions for each edge in the causal graph.

GitHub Repo: `https://github.com/nathanwang000/Shapley-Flow`

Discussion: Shapley Flow attribution analysis carries a lot of information about both the structure of the relationship between variables and its effect of particular groups of variables (explanation boundaries) on the predictions. On the rather disadvantageous side is that it requires knowledge of the dependency structure in the form of a directed causal graph, which limits the number of problems in which it can be applied. For readability reasons, it is limited to small numbers of variables. Also it requires definition of a background case, i.e. reference observation. Potential explanations may vary depending on the reference observation chosen.

2.13 Textual Explanations of Visual Models

Idea: The generation of textual descriptions of images is addressed by several machine learning models that contain both a part that processes the input images - typically a convolutional neural network (CNN) - and one that learns an adequate text sequence, usually a recurrent neural network (RNN). Those two parts cooperate for the production of image descriptive sentences that presupposes that a classification task is successfully accomplished. One of the first benchmark datasets that contained image descriptions was already invented in 2014, the Microsoft COCO (MS-COCO) [48]. The models that achieve a good performance classification, first detect components and concepts of the image and then construct sentences where objects, subjects as well as their characteristics are connected by verbs. The problem of semantic enrichment of images for language-related tasks is addressed in a number of ways (see, for example, the Visual Genome project [45]); however, in most cases, such descriptions are not directly tied to visual recognition tasks. Nevertheless, an advantage is that textual descriptions are easier to analyze and validate than attribution maps.

It is important to note that the mere description of the image's content is not equivalent to an explanation of the decision-making process of the neural network model. Unless the produced sentences contain the unique attributes that

help differentiate between the images of each class, the content of the words should not be considered class-relevant content. A solution to this problem is proposed in [31]. This method's main goal is to do exactly that; to find those characteristics that are discriminative, since they were used by the neural network models to accomplish the task - those exactly need to be present in the generated text. To achieve this, the training does not just use the relevance loss, which generates descriptions relevant to the predicted class based on context borrowed from a fine-grained image recognition model through conditional probabilities. A discriminative loss is invented to generate sentences rich in class-discriminative features. The introduced weight update procedure consists of two components, one based on the gradient of relevance loss and the second based on the gradient of discriminative loss, so that descriptions that are both relevant to the predicted class and contain words with high discriminative capacity are rewarded. The reinforcement learning method REINFORCE [85] is used for the backpropagation of the error through sampling during the training process.

GitHub Repo: `https://github.com/LisaAnne/ECCV2016`

Discussion: High METEOR [13] and CIDEr [82] scores for relevant explanations were measured for the generated sentences. It is necessary to compare the resulting explanations with experts since they only know the difference between sentences that correctly describe the visual content and ones that concentrate on what occurs only in the class the images belong. This is positive and negative at the same time; unfortunately, there is no way to check how much of the generated explanation is consistent without domain knowledge. Furthermore, data artefacts can also influence both the performance and explanation quality negatively. Overall though, even ablation studies where parts of the model were tested separately, showed that the components individually had a higher performance than when trained alone. That indicates that the common training of visual processing and textual explanation generation is beneficial for each part individually.

2.14 Integrated Gradients

Idea: The Integrated Gradients method [80] is based on two fundamental axioms, sensitivity and implementation invariance. Sensitivity means that non-zero attributions are given to every input and baseline that differ in one feature but have different predictions. Implementation invariance means that if two models behave identical/are functionally equivalent, then attributions must be identical. Although these two axioms sound very natural, it turns out that many attribution methods do not have these properties. In particular, when a model has flattened predictions for a specific point of interest, the gradient in the point of interest zeroes out and does not carry information useful for the explanation.

The approach proposed by the Integrated Gradients method for model f aggregates the gradients $\frac{\partial f(x)}{\partial x_i}$ computed along the path connecting the point of

interest x to the highlighted observation - the baseline x^* (for computer vision this could be a black image and for text an empty sentence).

More formally, for ith feature, Integrated Gradients are defined as

$$IntegratedGrads_i(x) = (x_i - x_i^*) \int_{\alpha=0}^{1} \frac{\partial f(x^* + \alpha(x - x^*))}{\partial x_i} d\alpha.$$

The integral can be replaced by a sum over a set of alpha values in the interval [0,1].

GitHub Repo: `https://github.com/ankurtaly/Integrated-Gradients`

Discussion: Integrated Gradients is a widespread technique for explaining deep neural networks or other differentiable models. It is a theoretically sound approach based on two desirable properties: sensitivity and implementation invariance. In addition, it is computationally efficient and uses gradient information at a few selected points α. The three main drawbacks are: (1) need for the baseline observation, selection of which significantly influence the attributions, (2) works only for differentiable models, suitable for neural networks but not, e.g., for decision trees, (3) by default, gradients are integrated along the shortest path between the baseline and the point of interest. Depending on the topology of the data, this path does not always make sense and cover the data. Furthermore, deep models usually suffer from the gradient shattering problem [12], which may negative affect the explanation (see discussion in [70]). Extensions to this method are proposed to overcome the above drawbacks.

2.15 Causal Models

Description: In the work of Madumal et al. [53] a structural causal model [29] is learned, which can be considered an extension of Bayesian Models [44], [71] of the RL environment with the use of counterfactuals. It takes into account events that would happen or environment states that would be reached under different actions taken by the RL agent. Ultimately, the goal of any RL agent is to maximize a long-term reward; the explanation provides causal chains until the reward receiving state is reached. The researchers pay attention to keep the explanations minimally complete, by removing some of the intermediate nodes in the causal chains, to conform to the explanation satisfaction conditions according to the Likert scale [33]. The counterfactual explanation is computed by comparing causal chain paths of actions not chosen by the agent (according to the trained policy). To keep the explanation as simple as possible, only the differences between the causal chains comprise the returned counterfactual explanation.

GitHub Repo: No Github Repo

Discussion: Model-free reinforcement learning with a relatively small state and action space has the advantage that we can explain how the RL agent takes its decisions in a causal way; since neural networks base their decisions on correlations, this is one of the first works towards causal explanations. The user has the ability to get answers to the questions "why" and "why not" an action was chosen by the agent. The provided explanations are appropriate according to satisfiability, ethics requirements and are personalized to the human mental model by the use of a dedicated user interface. On the rather negative side is that this explanation method is evaluated on problems with very small state and action space (9 and 4 correspondingly). Current RL problems have much larger state and action spaces and the solution can be found with the use of Deep Reinforcement Learning [27], [32]. The reason is, that the structural model of the environment dynamics are not known a priori and must be discovered and approximated during exploration. Furthermore, this work applies only to the finite domain, although the authors note that it will be part of their research work to extend it to continuous spaces.

2.16 Meaningful Perturbations

Idea: This approach was proposed by Fong and Vedaldi [21] and can be regarded as model-agnostic, perturbation-based explanation method. Thus, the explanation is computed solely based on the reaction of the model to a perturbed (or occluded) input sample. For a given sample $\mathbf{x}$, the method aims to synthesize a sparse occlusion map (i.e., the explanation) that leads to the maximum drop of the model's prediction $f(\mathbf{x})$, relative do the original prediction with the unperturbed $\mathbf{x}$. Thus, compared to simple occlusion-based techniques which naively perturb a given sample by sequentially occluding parts of it, the Meaningful Perturbation algorithm aims to directly learn the explanation by formulating the explanation problem as a meta-prediction task and using tools from optimization to solve it. Sparsity constraints ensure that the search focuses on finding the smallest possible perturbation mask that has the larger effect on the certainty of the classification performance.

GitHub Repo: `https://github.com/ruthcfong/perturb_explanations`

Discussion: As other model agnostic approaches, Meaningful Perturbations is a very flexible method, which can be directly applied to any machine learning model. The approach can be also interpreted from a rate-distortion perspective [43]. Since the Meaningful Perturbations method involves optimization, it is computationally much more demanding than than propagation-based techniques such as LRP. Also it is well-known that the perturbation process (occlusion or deletion can be seen as a particular type of perturbation), moves the sample out of the manifold of natural images and thus can introduce artifacts. The use of generative models have been suggested to overcome this out-of-manifold problem [1].

2.17 EXplainable Neural-Symbolic Learning (X-NeSyL)

Idea: Symbolic AI is an emerging field that has been shown to contribute immensely to Explainable AI. Neuro-Symbolic methods [24] incorporate prior human knowledge for various tasks such as concept learning and at the same time they produce output that is more interpretable, such as mathematical equations or Domain-Specific Languages (DSL) [55].

A research work that is dedicated to using symbolic knowledge of the domain experts, expressed as a knowledge graph (KG), to align it with the explanations of a neural network is the EXplainable Neural-Symbolic Learning (X-NeSyL) [20]. The researchers start with the goal to encourage the neural network that performs classification to assign feature importances to the object's parts in a way that corresponds to the compositional way humans classify. After using state-of-the-art CNN architectures and applying methods such as SHAP (see Section 2.9) to quantify the positive and negative influence of each detected feature, a graph is built that encompasses constraints and relations elicited from the computed importances. This graph is compared to the KG provided by human experts. A designated loss that punishes non-overlap between these two has been shown to boost explainability and in some cases performance.

GitHub Repo: `https://github.com/JulesSanchez/X-NeSyL`, `https://github.com/JulesSanchez/MonuMAI-AutomaticStyleClassification`

Discussion: This method can be seen as an explainability-by-design approach. That means that at each step of the training process, it is made sure that the end result will be interpretable. This is not an ad-hoc method; the training contains a loss to guide the neural network towards explanations that have a human-expert like structure. Furthermore, SHAP values provide intermediate feature relevance results that are straightforward to understand. Disadvantageous is that the same thing that fosters explainability, contributes to the negatives of this approach, namely that it needs domain-specific knowledge. This is not always easy to gather, it may be contradicting if several experts are involved and in that way constraints the network to compute in a specific way. The researchers comment on that particular issue and exercise their method with many datasets, test several CNN architectures and provide performance results with established as well as newly invented methods to see where and how the human-in-the-loop [37] works in practice.

3 Conclusion and Future Outlook

In the future, we expect that the newly invented xAI methods will capture causal dependencies. Therefore, it will be important to measure the quality of explanations so that an xAI method achieves a certain level of causal understanding [65] for a user with effectiveness, efficiency and satisfaction in a given context of use

[34]. Successful xAI models in the future will also require new human-AI interfaces [36] that enable contextual understanding and allow a domain expert to ask questions and counterfactuals [35] ("what-if" questions). This is where a human-in-the-loop can (sometimes - not always, of course) bring human experience and conceptual knowledge to AI processes [37]. Such conceptual understanding is something that the best AI algorithms in the world (still) lack, and this is where the international xAI community will make many valuable contributions in the future.

4 Acknowledgements

The authors declare no conflict of interests. This work does not raise any ethical issues. The content is partly based on the XAI seminar LV 706.315 provided by Andreas Holzinger since 2015 at Graz University of Technology. Parts of this work have been funded by the Austrian Science Fund (FWF), Project: P-32554, explainable AI, by the German Ministry for Education and Research (BMBF) through BIFOLD (refs. 01IS18025A and 01IS18037A), by the European Union's Horizon 2020 programme (grant no. 965221), Project iToBoS, and by the German Research Foundation (DFG), Emmy Noether Grant 437611051.

References

1. Agarwal, C., Nguyen, A.: Explaining image classifiers by removing input features using generative models. In: Proceedings of the Asian Conference on Computer Vision (2020)
2. Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K.T., Montavon, G., Samek, W., Müller, K.R., Dähne, S., Kindermans, P.J.: iNNvestigate neural networks! Journal of machine learning research (JMLR) **20**(93), 1–8 (2019)
3. Ali, A., Schnake, T., Eberle, O., Montavon, G., Müller, K.R., Wolf, L.: XAI for transformers: Better explanations through conservative propagation. arXiv preprint arXiv:2202.07304 (2022)
4. Anders, C.J., Neumann, D., Samek, W., Müller, K.R., Lapuschkin, S.: Software for dataset-wide XAI: From local explanations to global insights with Zennit, CoRelAy, and ViRelAy. arXiv preprint arXiv:2106.13200 (2021)
5. Anders, C.J., Weber, L., Neumann, D., Samek, W., Müller, K.R., Lapuschkin, S.: Finding and removing clever hans: Using explanation methods to debug and improve deep models. Information Fusion **77**, 261–295 (2022)
6. Arras, L., Arjona-Medina, J., Widrich, M., Montavon, G., Gillhofer, M., Müller, K.R., Hochreiter, S., Samek, W.: Explaining and interpreting lstms. In: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Lecture Notes in Computer Science, vol. 11700, pp. 211–238. Springer (2019)
7. Arras, L., Montavon, G., Müller, K.R., Samek, W.: Explaining recurrent neural network predictions in sentiment analysis. In: Proceedings of the EMNLP'17 Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis (WASSA). pp. 159–168. Association for Computational Linguistics (2017)
8. Arras, L., Osman, A., Samek, W.: Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations. Information Fusion **81**, 14–40 (2022)

9. Asif, N.A., Sarker, Y., Chakrabortty, R.K., Ryan, M.J., Ahamed, M.H., Saha, D.K., Badal, F.R., Das, S.K., Ali, M.F., Moyeen, S.I.: Graph neural network: A comprehensive review on non-euclidean space. IEEE Access **9**, 60588–60606 (2021)
10. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**(7), e0130140 (2015)
11. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.R.: How to explain individual classification decisions. Journal of Machine Learning Research **11**, 1803–1831 (2010)
12. Balduzzi, D., Frean, M., Leary, L., Lewis, J., Ma, K.W.D., McWilliams, B.: The shattered gradients problem: If resnets are the answer, then what is the question? In: International Conference on Machine Learning. pp. 342–350. PMLR (2017)
13. Banerjee, S., Lavie, A.: Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In: Proceedings of the ACL workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. pp. 65–72 (2005)
14. Becking, D., Dreyer, M., Samek, W., Müller, K., Lapuschkin, S.: Ecq^{x}: Explainability-driven quantization for low-bit and sparse dnns. In: xxAI - Beyond explainable Artificial Intelligence. Lecture Notes in Computer Science, Springer (2022)
15. Bengio, Y., Lecun, Y., Hinton, G.: Deep learning for ai. Communications of the ACM **64**(7), 58–65 (2021)
16. Biecek, P.: DALEX: Explainers for Complex Predictive Models in R. Journal of Machine Learning Research **19**(84), 1–5 (2018), `http://jmlr.org/papers/v19/18-416.html`
17. Biecek, P., Burzykowski, T.: Explanatory Model Analysis. Chapman and Hall/CRC, New York (2021), `https://pbiecek.github.io/ema/`
18. Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R., Samek, W.: Layer-wise relevance propagation for neural networks with local renormalization layers. Artificial Neural Networks and Machine Learning - ICANN 2016 **9887**, 63–71 (2016)
19. Clough, J.R., Oksuz, I., Puyol-Antón, E., Ruijsink, B., King, A.P., Schnabel, J.A.: Global and local interpretability for cardiac mri classification. In: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI). pp. 656–664. Springer (2019)
20. Díaz-Rodríguez, N., Lamas, A., Sanchez, J., Franchi, G., Donadello, I., Tabik, S., Filliat, D., Cruz, P., Montes, R., Herrera, F.: Explainable neural-symbolic learning (x-nesyl) methodology to fuse deep learning representations with expert knowledge graphs: the monumai cultural heritage use case. arXiv preprint arXiv:2104.11914 (2021)
21. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE international conference on computer vision. pp. 3429–3437 (2017)
22. Frye, C., de Mijolla, D., Cowton, L., Stanley, M., Feige, I.: Shapley-based explainability on the data manifold. arXiv preprint arXiv:2006.01272 (2020)
23. Frye, C., Rowat, C., Feige, I.: Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1229–1239 (2020)
24. Garcez, A.S.d., Broda, K.B., Gabbay, D.M.: Neural-symbolic learning systems: foundations and applications. Springer Science & Business Media (2012)

25. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (MA) (2016)
26. Gosiewska, A., Biecek, P.: iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models. arXiv preprint arXiv:1903.11420 (2019)
27. Graesser, L., Keng, W.L.: Foundations of deep reinforcement learning: theory and practice in Python. Addison-Wesley Professional (2019)
28. Gunning, D., Aha, D.W.: Darpa's explainable artificial intelligence program. AI Magazine **40**(2), 44–58 (2019)
29. Halpern, J.Y., Pearl, J.: Causes and explanations: A structural-model approach. part ii: Explanations. The British Journal for the Philosophy of Science **56**(4), 889–911 (2005)
30. Hedström, A., Weber, L., Bareeva, D., Motzkus, F., Samek, W., Lapuschkin, S., Höhne, M.M.C.: Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations. arXiv preprint arXiv:2202.06861 (2022)
31. Hendricks, L.A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., Darrell, T.: Generating visual explanations. In: European Conference on Computer Vision. pp. 3–19. Springer (2016)
32. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems **33**(6), 750–797 (2019)
33. Hoffman, R.R., Mueller, S.T., Klein, G., Litman, J.: Metrics for explainable ai: Challenges and prospects. arXiv preprint arXiv:1812.04608 (2018)
34. Holzinger, A., Carrington, A., Mueller, H.: Measuring the quality of explanations: The system causability scale (scs). comparing human and machine explanations. KI - Künstliche Intelligenz (German Journal of Artificial intelligence), Special Issue on Interactive Machine Learning, Edited by Kristian Kersting, TU Darmstadt **34**(2), 193–198 (2020)
35. Holzinger, A., Malle, B., Saranti, A., Pfeifer, B.: Towards multi-modal causability with graph neural networks enabling information fusion for explainable ai. Information Fusion **71**(7), 28–37 (2021)
36. Holzinger, A., Mueller, H.: Toward human-ai interfaces to support explainability and causability in medical ai. IEEE Computer **54**(10), 78–86 (2021)
37. Holzinger, A., Plass, M., Kickmeier-Rust, M., Holzinger, K., Crisan, G.C., Pintea, C.M., Palade, V.: Interactive machine learning: experimental evidence for the human in the algorithmic loop. Applied Intelligence **49**(7), 2401–2414 (2019)
38. Huang, Q., Yamada, M., Tian, Y., Singh, D., Yin, D., Chang, Y.: GraphLIME: Local interpretable model explanations for graph neural networks. arXiv preprint arXiv:2001.06216v1 (2020)
39. Kauffmann, J., Esders, M., Montavon, G., Samek, W., Müller, K.R.: From clustering to cluster explanations via neural networks. arXiv preprint arXiv:1906.07633 (2019)
40. Kauffmann, J., Müller, K.R., Montavon, G.: Towards explaining anomalies: a deep taylor decomposition of one-class models. Pattern Recognition **101**, 107198 (2020)
41. Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al.: Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In: International Conference on Machine Learning. pp. 2668–2677. PMLR (2018)
42. Kohlbrenner, M., Bauer, A., Nakajima, S., Binder, A., Samek, W., Lapuschkin, S.: Towards best practice in explaining neural network decisions with LRP. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2020)

43. Kole, S., Bruna, J., Kutyniok, G., Levie, R., Nguyen, D.A.: A rate-distortion framework for explaining neural network decisions. In: xxAI - Beyond explainable Artificial Intelligence. Lecture Notes in Computer Science, Springer (2022)
44. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT Press (2009)
45. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision **123**(1), 32–73 (2017)
46. Lapuschkin, S., Binder, A., Müller, K.R., Samek, W.: Understanding and comparing deep neural networks for age and gender classification. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 1629–1638 (2017)
47. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.R.: Unmasking clever hans predictors and assessing what machines really learn. Nature Communications **10**, 1096 (2019)
48. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
49. Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I.: From local explanations to global understanding with explainable ai for trees. Nature Machine Intelligence **2**(1), 56–67 (2020)
50. Lundberg, S.M., Erion, G.G., Lee, S.I.: Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018)
51. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems **30**, 4765–4774 (2017)
52. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. Advances in neural information processing systems **33**, 19620–19631 (2020)
53. Madumal, P., Miller, T., Sonenberg, L., Vetere, F.: Explainable reinforcement learning through a causal lens. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 2493–2500 (2020)
54. Mamalakis, A., Ebert-Uphoff, I., Barnes, E.: Explainable artificial intelligence in meteorology and climate science: Model fine-tuning, calibrating trust and learning new science. In: xxAI - Beyond explainable Artificial Intelligence. Lecture Notes in Computer Science, Springer (2022)
55. Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B., Wu, J.: The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. arXiv preprint arXiv:1904.12584 (2019)
56. Mittelstadt, B.: Principles alone cannot guarantee ethical ai. Nature Machine Intelligence **1**, 1–7 (2019)
57. Molnar, C., König, G., Herbinger, J., Freiesleben, T., Dandl, S., Scholbeck, C.A., Casalicchio, G., Grosse-Wentrup, M., Bischl, B.: Pitfalls to avoid when interpreting machine learning models. arXiv preprint arXiv:2007.04131 (2020)
58. Montavon, G., Binder, A., Lapuschkin, S., Samek, W., Müller, K.R.: Layer-wise relevance propagation: An overview. In: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Lecture Notes in Computer Science, vol. 11700, pp. 193–209. Springer (2019)

59. Montavon, G., Kauffmann, J., Samek, W., Müller, K.R.: Explaining the predictions of unsupervised learning models. In: xxAI - Beyond explainable Artificial Intelligence. Lecture Notes in Computer Science, Springer (2022)
60. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017)
61. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital Signal Processing **73**, 1–15 (2018)
62. Morch, N.J., Kjems, U., Hansen, L.K., Svarer, C., Law, I., Lautrup, B., Strother, S., Rehm, K.: Visualization of neural networks using saliency maps. In: Proceedings of ICNN'95-International Conference on Neural Networks. vol. 4, pp. 2085–2090 (1995)
63. O'Sullivan, S., Nevejans, N., Allen, C., Blyth, A., Leonard, S., Pagallo, U., Holzinger, K., Holzinger, A., Sajid, M.I., Ashrafian, H.: Legal, regulatory, and ethical frameworks for development of standards in artificial intelligence (ai) and autonomous robotic surgery. The International Journal of Medical Robotics and Computer Assisted Surgery **15**(1), e1968 (2019)
64. Pearl, J.: The limitations of opaque learning machines. In: Brockman, J. (ed.) Possible Minds: 25 Ways of Looking at AI, pp. 13–19. Penguin, New York (2019)
65. Pearl, J.: The seven tools of causal inference, with reflections on machine learning. Communications of the ACM **62**(3), 54–60 (2019)
66. Pearl, J., Mackenzie, D.: The book of why. Basic Books, New York (2018)
67. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016). pp. 1135–1144. ACM (2016)
68. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (2018)
69. Robnik-Šikonja, M., Kononenko, I.: Explaining classifications for individual instances. IEEE Transactions on Knowledge and Data Engineering **20**(5), 589–600 (2008)
70. Samek, W., Montavon, G., Lapuschkin, S., Anders, C.J., Müller, K.R.: Explaining deep neural networks and beyond: A review of methods and applications. Proceedings of the IEEE **109**(3), 247–278 (2021)
71. Saranti, A., Taraghi, B., Ebner, M., Holzinger, A.: Insights into learning competence through probabilistic graphical models. In: International cross-domain conference for machine learning and knowledge extraction. pp. 250–271. Springer (2019)
72. Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K.T., Müller, K.R., Montavon, G.: XAI for graphs: Explaining graph neural network predictions by identifying relevant walks. arXiv preprint arXiv:2006.03589 (2020)
73. Schneeberger, D., Stoeger, K., Holzinger, A.: The european legal framework for medical ai. In: International Cross-Domain Conference for Machine Learning and Knowledge Extraction, Springer LNCS 12279, pp. 209–226. Springer, Cham (2020)
74. Schoelkopf, B.: Causality for machine learning. arXiv preprint arXiv:1911.10500 (2019)
75. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
76. Stoeger, K., Schneeberger, D., Kieseberg, P., Holzinger, A.: Legal aspects of data cleansing in medical ai. Computer Law and Security Review **42**, 105587 (2021)

77. Student: The probable error of a mean. Biometrika pp. 1–25 (1908)
78. Sturm, I., Lapuschkin, S., Samek, W., Müller, K.R.: Interpretable deep neural networks for single-trial eeg classification. Journal of Neuroscience Methods **274**, 141–145 (2016)
79. Sun, J., Lapuschkin, S., Samek, W., Binder, A.: Explain and improve: Lrp-inference fine tuning for image captioning models. Information Fusion **77**, 233–246 (2022)
80. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic Attribution for Deep Networks. In: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 3319–3328. PMLR (06–11 Aug 2017)
81. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press (2018)
82. Vedantam, R., Lawrence Zitnick, C., Parikh, D.: Cider: Consensus-based image description evaluation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4566–4575 (2015)
83. Vu, M., Thai, M.T.: Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. Advances in Neural Information Processing Systems **33**, 12225–12235 (2020)
84. Wang, J., Wiens, J., Lundberg, S.: Shapley Flow: A Graph-based Approach to Interpreting Model Predictions. In: 24th International Conference on Artificial Intelligence and Statistics (AISTATS). Proceedings of Machine Learning Research, vol. 130, pp. 721–729. PMLR (2021)
85. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**(3), 229–256 (1992)
86. Xu, J., Xue, M., Picek, S.: Explainability-based backdoor attacks against graph neural networks. In: Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning. pp. 31–36 (2021)
87. Yeom, S.K., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K.R., Samek, W.: Pruning by explaining: A novel criterion for deep neural network pruning. Pattern Recognition **115**, 107899 (2021)
88. Yuan, H., Tang, J., Hu, X., Ji, S.: Xgnn: Towards model-level explanations of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 430–438 (2020)
89. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. pp. 818–833. Springer (2014)
90. Zhang, A., Lipton, Z.C., Li, M., Smola, A.J.: Dive into deep learning. Release 0.17.0, Open Source (2021)
91. Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M.: Visualizing deep neural network decisions: Prediction difference analysis. arXiv preprint arXiv:1702.04595 (2017)