# Higher-Order Explanations of Graph Neural Networks via Relevant Walks

Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T. Schütt, Klaus-Robert Müller, *Member, IEEE*, and Grégoire Montavon

**Abstract**—Graph Neural Networks (GNNs) are a popular approach for predicting graph structured data. As GNNs tightly entangle the input graph into the neural network structure, common explainable AI approaches are not applicable. To a large extent, GNNs have remained black-boxes for the user so far. In this paper, we show that GNNs can in fact be naturally explained using *higher-order* expansions, i.e., by identifying groups of edges that jointly contribute to the prediction. Practically, we find that such explanations can be extracted using a nested attribution scheme, where existing techniques such as layer-wise relevance propagation (LRP) can be applied at each step. The output is a collection of walks into the input graph that are relevant for the prediction. Our novel explanation method, which we denote by GNN-LRP, is applicable to a broad range of graph neural networks and lets us extract practically relevant insights on sentiment analysis of text data, structure-property relationships in quantum chemistry, and image classification.

**Index Terms**—Graph neural networks, higher-order explanations, layer-wise relevance propagation, explainable machine learning

## 1 Introduction

Many interesting structures found in scientific and industrial applications can be expressed as graphs. Examples are lattices in fluid modeling, molecular geometry, biological interaction networks, or social/historical networks. Graph neural networks (GNNs) [1], [2] have been proposed as a method to learn from observations in general graph structures and have found use in an ever growing number of applications [3], [4], [5], [6], [7], [8]. While GNNs make useful predictions, they typically act as black-boxes, and it has neither been directly possible (1) to extract novel insight from the learned model nor (2) to verify that the model has made the intended use of the graph structure, e.g., that it has avoided Clever Hans phenomena [9].

Explainable AI (XAI) is an emerging research area that aims to extract interpretable insights from trained ML models [10], [11]. So far, research has focused, for example, on full black-box models [12], [13], self-explainable models [14], [15], or deep neural networks [16], where in all cases, the prediction can be attributed to the input features. For a GNN, however, the graph being received as input is deeply entangled with the model itself, hence requiring a more sophisticated approach.

In this paper, we propose a theoretically founded XAI method for explaining GNN predictions. The conceptual starting point of our method is the observation that the function implemented by the GNN is locally polynomial with the input graph. This function can therefore be analyzed using a *higher-order* Taylor expansion to arrive at an attribution of the GNN prediction on collections of edges, e.g., *walks* into the input graph. —Such an attribution scheme goes beyond existing XAI techniques for GNNs that are limited to identifying individual nodes or edges.

Furthermore, we find that the higher-order expansion can be expressed as a nesting of multiple first-order expansions, starting at the top layer of the GNN and moving towards the input layer. This theoretical insight enables a principled adaptation of the Layer-wise Relevance Propagation (LRP) [16] explanation technique to GNN models, where the propagation procedure is guided along individual walks in the input graph. The resulting procedure that we propose and that we denote by GNN-LRP is shown in Fig. 1.

GNN-LRP applies directly to a broad range of GNN architectures, without need to learn a surrogate function, nor to run any optimization procedure. We demonstrate


- *Thomas Schnake, Oliver Eberle, Kristof T. Schütt, and Grégoire Montavon are with the BIFOLD – Berlin Institute for the Foundations of Learning and Data, Berlin Institute of Technology (TU Berlin), 10587 Berlin, Germany. E-mail: {t.schnake, kristof.schuett, gregoire.montavon}@tu-berlin.de, oliver.eberle@campus.tu-berlin.de.*
- *Jonas Lederer is with the Berlin Institute of Technology (TU Berlin), 10587 Berlin, Germany. E-mail: jonas.lederer@tu-berlin.de.*
- *Shinichi Nakajima is with the BIFOLD – Berlin Institute for the Foundations of Learning and Data, Berlin Institute of Technology (TU Berlin), 10587 Berlin, Germany, and also with the RIKEN AIP, Tokyo 103-0027, Japan. E-mail: nakajima@tu-berlin.de.*
- *Klaus-Robert Müller is with the Google Research, Brain team, Berlin, BIFOLD – Berlin Institute for the Foundations of Learning and Data, the Berlin Institute of Technology (TU Berlin), 10587 Berlin, Germany, and with the Department of Artificial Intelligence, Korea University, Seoul 136-713, Korea, and also with the Max Planck Institut für Informatik, 66123 Saarbrücken, Germany. E-mail: klaus-robert.mueller@tu-berlin.de.*


*Manuscript received 26 November 2020; revised 21 September 2021; accepted 22 September 2021. Date of publication 24 September 2021; date of current version 3 October 2022.*
*This work was funded by the German Ministry for Education and Research as BIFOLD – Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A), and the German Research Foundation (DFG) as Math+: Berlin Mathematics Research Center (EXC 2046/1, project-ID: 390685689). This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea Government (No. 2019-0-00079, Artificial Intelligence Graduate School Program, Korea University).*
*(Corresponding authors: Grégoire Montavon, Klaus-Robert Müller.)*
*Recommended for acceptance by S. Ji.*
*Digital Object Identifier no. 10.1109/TPAMI.2021.3115452*

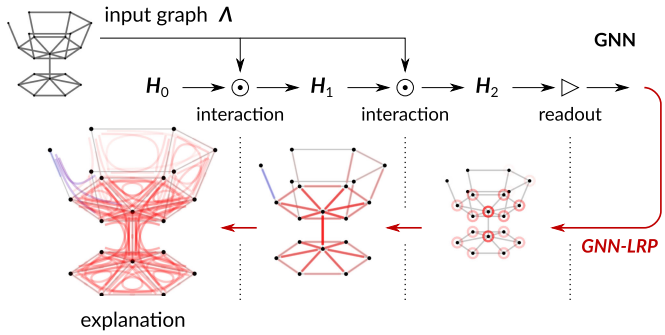


Fig. 1. High-level illustration of GNN-LRP. The explanation procedure starts at the GNN output, and proceeds backwards to progressively uncover the walks that are relevant for the prediction.

GNN-LRP on a variety of GNN models from diverse application fields: (1) a sentiment prediction model receiving sentence parse trees as input, (2) a state-of-the-art GNN for quantum mechanically accurate prediction of electronic properties from molecular graphs, and (3) a widely adopted image classifier that we view as a GNN operating on pixel lattices.—For each GNN model, our explanation method produces detailed and reliable explanations of the decision strategy from which novel application insights can be obtained.

The code for this paper can be found at https://git.tu-berlin.de/thomas_schnake/paper_gnn_lrp.

### 1.1 Related Work

We focus here on the related work that most directly connects to our novel GNN explanation approach, in particular, (1) explanation techniques based on higher-order analysis, and (2) explanation techniques that are specialized for GNNs. For a more comprehensive set of related works, we refer the reader to the review papers [17], [18] for XAI and [2], [19] for GNNs.

#### 1.1.1 Higher-Order Explanations

Second-order methods (e.g., based on the model's Hessian) have been proposed to attribute predictions to pairs of input features [20], [21], [22]. Another work [14] incorporates an explicit sum-of-interactions structure into the model, in order to obtain second-order or higher-order explanations. Another approach [23] detects higher-order feature interaction with an iterative algorithm which inspects neural network weights at the different layers.

Our work proposes instead to use the framework of Taylor expansions to arrive in a principled manner to the higher-order explanations, and it identifies GNNs as an important use case for such explanations.

#### 1.1.2 Explaining Graph Neural Networks

The work [24] extends explanation techniques such as *Grad-CAM* or *Excitation Backprop* to the GNN model, and arrives at an attribution on nodes of the graph. In an NLP context, graph convolutional networks (GCNs) have been explained in terms of nodes and edges in the input graph using the *LRP* explanation method [25]. *GNNExplainer* [26] and *PGExplainer* [27] explain the model by extracting the subgraph that maximizes the mutual information to the prediction for the original graph. *XGNN* [28] is a model-level explanation method, which produces typical graphs for a target class. *PGMExplainer* [29] learns a probabilistic graphical model from the network, which is then able to measure the probability for different higher order feature interactions in the model. *GraphMask* [30] learns binary masks for edges in each layer to retain only the connections which are most important for the prediction. *Gem* [31] is a trained generative model that generates causal explanations of GNNs. The method *SubgraphX* [32] proposes a Monte-Carlo tree search to find relevant subgraphs in the input graph and uses Shapley values as an attribution function for subgraphs. Other recently proposed methods that map the GNN prediction to graph substructures include *Trap2* [33] and *GraphLime* [34]. In [35] the authors present a survey of different interpretation methods for GNNs.

Regarding the quality of the relevance features, most of the proposed methods attribute the GNN prediction to nodes or edges of the input graph [24], [26], [27], whereas GNN-LRP gives scores for higher-order features, such as *sequences* of edges. The quality of the attributions for GraphMask and PGMExplainer are comparable with our approach. Yet, both methods learn to understand the prediction strategy of a GNN by a given optimization criterion, where the reliability of such explanations strongly depends on the well-posedness of the optimization criteria (e.g., convexity). Instead our method is independent of any additional optimization criterion. The method SubgraphX [32] attributes the prediction to subgraphs of the input graph and is closely related to our subgraph selection technique presented in Section 4.1. SubgraphX uses a Monte-Carlo optimization algorithm to find the most relevant subgraph, whereas we use either a *local best guess* or a *random sampling* approach (cf. Appendix D of the Supplement, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3115452). SubgraphX uses Shapley values for the subgraph scoring, whereas we use a backward propagation pass.

## 2 Towards Explaining GNNs

In this section, we briefly introduce graph neural networks and then develop a Taylor-based explanation framework for explaining these models. Throughout the paper, we make use of graph-specific notation that we summarize in Table 1.

### 2.1 Graph Neural Networks

Graph neural networks (GNNs) [1], [2] are special types of neural networks that receive a graph as input. In practice, graphs can take a variety of forms, e.g., directed, undirected, labeled, unlabeled, spatial, time-evolving, etc. To handle the high heterogeneity of graph structures, many variants of GNNs have been developed (e.g., [36], [37], [38]). One commonality of most GNNs, however, is that the input graph is not located at the first layer, but occurs instead at multiple layers, by defining the connectivity of the network itself.

Graph neural networks are typically constructed by stacking several *interaction blocks*. Each block $t = 1 \ldots T$ computes a graph representation $\boldsymbol{H}_t \in \mathbb{R}^{n \times d_t}$ where $n$ is the number of nodes in the input graph and $d_t$ is the number of dimensions used to represent each node. Within a block, the representation is produced by applying (i) an *aggregate* step where each node receives information from the neighboring

TABLE 1
Notation Used Throughout the Paper

| | |
|---|---|
| $\boldsymbol{H}, \boldsymbol{X}$ | (uppercase and bold) Matrix or tensor. |
| $h, x$ | (lowercase) Vector or scalar. |
| $\odot$ and $\oslash$ | Element-wise multiplication and division respectively. |
| $\langle \cdot, \cdot \rangle$ | Euclidean scalar product. |
| $\mathcal{G}$ | A graph defined by sets of nodes and edges. |
| $\mathcal{N}$, $\mathcal{E}$ and $\mathcal{B}$ | Node, edge and bag-of-edges in a graph respectively. |
| $\mathcal{W}$ | A walk as an ordered sequence of nodes. |
| $\mathcal{X}$ | Any graph feature, either an edge, node or bag-of-edges. |
| $\mathcal{X} \in \mathcal{G}$ | Set of graph features in a graph, where $\mathcal{X}$ can also be interchanged by $\mathcal{N}$, $\mathcal{E}$ or $\mathcal{B}$. |
| $\mathcal{W} \in \mathcal{B}$ | Set of all walks composed by the bag-of-edges $\mathcal{B}$. |
| $J, K, \ldots$ | Node indices in a GNN. Adjacent letters in the alphabet indicate adjacent blocks in the GNN. |
| $a, b, \ldots$ | Neuron indices within the node. Adjacent letters in the alphabet indicate adjacent layers in the network. |
| $\boldsymbol{\Lambda}$, $\lambda_{\mathcal{E}}$, $\lambda_{JK}$ | $\boldsymbol{\Lambda}$ is the connectivity matrix of a GNN. For each edge $\mathcal{E}$ and node pair $(J, K)$, $\lambda_{\mathcal{E}}$ and $\lambda_{JK}$ define its corresponding connectivity weigtht respectively. |
| $\boldsymbol{H}_t$ | The hidden representation of a GNN at block $t$. |
| $\boldsymbol{H}_{t,I}$ | The representation of node $I$ at block $t$. |

*The table is separated between general mathematical notation (top), graph theoretic notation (middle) and notation that is used in the context of GNNs (bottom).*

nodes, and (ii) a *combine* step that extracts new features for each node. These two steps (cf. [39]) connect the representations $\boldsymbol{H}_{t-1}$ and $\boldsymbol{H}_t$ of consecutive blocks as

$$\text{aggregate:} \quad \boldsymbol{Z}_t = \boldsymbol{\Lambda}\boldsymbol{H}_{t-1} \tag{1}$$

$$\text{combine:} \quad \boldsymbol{H}_t = \big(\mathcal{C}_t(\boldsymbol{Z}_{t,K})\big)_K, \tag{2}$$

where $\boldsymbol{\Lambda}$ is the *input graph* given as a matrix of size $n \times n$, e.g., the adjacency matrix to which we add self-connections. We denote by $\boldsymbol{Z}_{t,K}$ the row of $\boldsymbol{Z}_t$ associated to node $K$, and $\mathcal{C}_t$ is a 'combine' function, typically a one-layer or multilayer neural network, that produces the new representation for each node in the graph.

The whole input-output relation implemented by the GNN can then be expressed as a function

$$f(\boldsymbol{\Lambda}; \boldsymbol{H}_0) = g\big(\boldsymbol{H}_T\big(\boldsymbol{\Lambda}, \boldsymbol{H}_{T-1}\big(\boldsymbol{\Lambda}, \ldots \boldsymbol{H}_1\big(\boldsymbol{\Lambda}, \boldsymbol{H}_0\big)\big)\big)\big), \tag{3}$$

which is a recursive application of Eqs. (1) and (2) starting from some *initial state* $\boldsymbol{H}_0 \in \mathbb{R}^{n \times d_0}$, followed by a readout function $g$. The initial state typically incorporates information that is intrinsic to the nodes, or it can be set to constant values if no such information is present. The readout function is typically a classifier (or regressor) of the whole graph, but it can also be chosen to apply to subsets of nodes, for example, for node classification or link prediction tasks [19], [40].

## 2.2 First-Order Explanation

Consider first a 'classical' approach to explanation where we attribute the output of the neural network to variables in the first layer. In the case of the GNN, the first layer is given by the initial state $\boldsymbol{H}_0$.

Let us now view the GNN as a function of the initial state, i.e., $f(\boldsymbol{H}_0)$. We will also denote by $\boldsymbol{H}_{0,I}$ the row of $\boldsymbol{H}_0$ associated to node $I$. A Taylor expansion of the function $f$ at some reference point $\widetilde{\boldsymbol{H}}_0$ gives

$$f(\boldsymbol{H}_0) = \sum_I \left\langle \frac{\partial f}{\partial \boldsymbol{H}_{0,I}}\bigg|_{\widetilde{\boldsymbol{H}}_0}, (\boldsymbol{H}_{0,I} - \widetilde{\boldsymbol{H}}_{0,I}) \right\rangle + \ldots, \tag{4}$$

where '$\ldots$' represents the zero-, second- and higher-order terms that have not been expanded, and where the sum represents the first-order terms. Because the sum runs over all nodes $I$ in the input graph, the addends in Eq. (4) readily provide an attribution of the GNN output to each node.

It is arguable, however, whether this attribution can truly be interpreted as identifying node contributions. Indeed, the attribution may only reflect the importance of a node in the first layer, and not in the higher layers. Furthermore, an attribution of the prediction on nodes may not be sufficient for the application needs. For example, it does not tell us whether a node is important by itself, or if it is important because of its connections to other nodes or some more complex structure in the graph.

These limitations can be attributed to the fact that we have performed the decomposition w.r.t. the initial state $\boldsymbol{H}_0$ instead of the 'true' input $\boldsymbol{\Lambda}$.

## 2.3 Higher-Order Explanation

Consider now the true input $\boldsymbol{\Lambda}$ of the GNN. Since it occurs in every interaction block and applies in a multiplicative manner (cf. Eq. (1)), a first-order analysis of the GNN function $f(\boldsymbol{\Lambda})$ would not be suitable to identify the multiplicative interactions. These interactions can however be identified by applying a *higher-order* Taylor expansion.

In the following, we will use the additional notation $\lambda_{\mathcal{E}}$ to denote the element of the matrix $\boldsymbol{\Lambda}$ associated to a particular edge $\mathcal{E}$ of the graph. Assuming that $f(\boldsymbol{\Lambda})$ is smooth on the relevant input domain, we can compute at some reference point $\widetilde{\boldsymbol{\Lambda}}$, a $T$-order Taylor expansion

$$f(\boldsymbol{\Lambda}) = \sum_{\mathcal{B}} \underbrace{\frac{1}{\alpha_{\mathcal{B}}!} \frac{\partial^T f}{\partial \lambda_{\mathcal{E}_1} \ldots \partial \lambda_{\mathcal{E}_T}}\bigg|_{\widetilde{\boldsymbol{\Lambda}}} \cdot \Delta_{\mathcal{E}_1} \cdot \ldots \cdot \Delta_{\mathcal{E}_T}}_{R_{\mathcal{B}}} + \ldots, \tag{5}$$

with $\Delta_{\mathcal{E}} := (\lambda_{\mathcal{E}} - \widetilde{\lambda}_{\mathcal{E}})$ and where we define $\alpha_{\mathcal{B}}! := \prod_{\mathcal{E}} \alpha_{\mathcal{B},\mathcal{E}}!$ with $\alpha_{\mathcal{B},\mathcal{E}}$ denoting the number of occurrences of edge $\mathcal{E}$ in the bag $\mathcal{B}$. The sum runs over all bags $\mathcal{B}$ of $T$ edges. Hence, the terms of the sum capture the joint effect of multiple edges on the GNN output. The last line '$+\ldots$' represents the non-expanded terms of order lower or higher than $T$. Anecdotally, for certain classes of functions, in particular, piecewise multilinear positively homogeneous functions (an example is

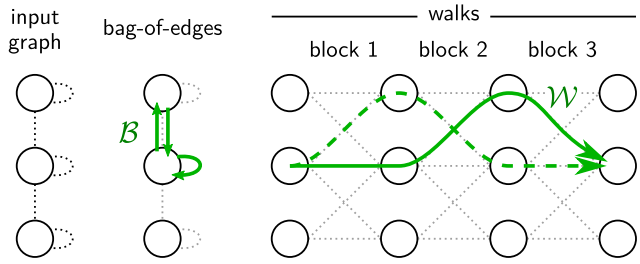


Fig. 2. Illustration of a bag-of-edges $\mathcal{B}$ and the corresponding walks $\mathcal{W}$ for a simple input graph of three nodes. The two walks associated to the given bag-of-edges are shown with a solid and a dashed line, respectively.

GNNs with ReLU nonlinearity and zero biases), choosing the reference point $\widetilde{\mathbf{\Lambda}} = s\mathbf{\Lambda}$ makes the non-expanded terms vanish in the limit of $s \to 0$, thereby leading to the conservation property $\sum_{\mathcal{B}} R_{\mathcal{B}} = f(\mathbf{\Lambda})$ (cf. Appendix A of the Supplement, available online).

### 2.4 Nested Computation and Relevant Walks

The higher-order Taylor expansion presented above is conceptually simple and mathematically founded. However, systematically extracting higher-order derivatives of a neural network is difficult and does not scale to complex models.

To address this first limitation, we introduce the concept of a *walk* $\mathcal{W}$, which we define to be an *ordered* sequence of nodes that are connected in consecutive blocks of the GNN. The relation between bag-of-edges and walks is illustrated for a simple graph in Fig. 2.

Because each walk maps to a particular bag-of-edges, a walk-based explanation inherits all information contained in the bag-of-edges explanation. In particular, it is always possible to recover the bag-of-edges explanation from a walk-based explanation by applying the pooling operation $R_{\mathcal{B}} = \sum_{\mathcal{W} \in \mathcal{B}} R_{\mathcal{W}}$. However, using walks brings two further advantages:

1) A walk-based explanation gives more information, how multiple blocks of the GNN have been used to arrive at the prediction. For example, as illustrated in Fig. 2, it can reveal whether message passing between two nodes has occurred in the first or in the last blocks of the GNN.
2) The fact that the walks connect to the structure of the GNN more directly, makes the explanation easier to compute.

To show this, we introduce the new variable $\mathbf{\Lambda}^\star \leftarrow (\mathbf{\Lambda}, \ldots, \mathbf{\Lambda})$ which distinguishes between edges occurring in different blocks of the GNN, and express the GNN output as a function of this expanded input, i.e., $f(\mathbf{\Lambda}^\star)$. We also adopt a node-based notation, where walks are given by the sequence of nodes they traverse from the first interaction block to the last one, e.g., $\mathcal{W} = (\ldots, J, K, L, \ldots)$. The letters $J, K, L$ denote nodes between the consecutive blocks, and '$\ldots$' acts as a placeholder for the leading and trailing nodes of the walk. We further denote by $\lambda^\star_{JK}$ the element of $\mathbf{\Lambda}^\star$ representing the connection between node $J$ and node $K$.

**Proposition 1.** *For the considered function $f(\mathbf{\Lambda}^\star)$ the higher-order terms $R_{\mathcal{B}}$ in Eq. (5) can be equivalently computed as a sequential application of a first order Taylor decomposition along a walk $\mathcal{W} = (\ldots, J, K, L, \ldots)$ at the root point $\widetilde{\mathbf{\Lambda}}^\star$ with $0 \preceq \widetilde{\mathbf{\Lambda}}^\star \prec \mathbf{\Lambda}^\star$*

$$R_{\mathcal{W}} = \frac{\partial}{\partial \ldots} \underbrace{\left( \frac{\partial}{\partial \lambda^\star_{JK}} \underbrace{\left( \frac{\partial \ldots}{\partial \lambda^\star_{KL}} \Big|_{\widetilde{\mathbf{\Lambda}}^\star} \cdot \Delta^\star_{KL} \right)}_{R_{KL\ldots}} \Big|_{\widetilde{\mathbf{\Lambda}}^\star} \cdot \Delta^\star_{JK} \right)}_{R_{JKL\ldots}} \Big|_{\widetilde{\mathbf{\Lambda}}^\star} \cdot \ldots, \tag{6}$$

*with $\Delta^\star_{\mathcal{E}} := (\lambda^\star_{\mathcal{E}} - \widetilde{\lambda}^\star_{\mathcal{E}})$, and then applying the pooling operation $R_{\mathcal{B}} = \sum_{\mathcal{W} \in \mathcal{B}} R_{\mathcal{W}}$.*

(A proof is given in Appendix B of the Supplement, available online.) In other words, the attribution can be produced by analyzing each block iteratively from the top of the network to the input features. Here again, when the function is a nesting of piecewise linear positively homogeneous functions, choosing the reference point $\widetilde{\mathbf{\Lambda}}^\star = s\mathbf{\Lambda}^\star$ gives an explanation that is conservative in the limit of $s \to 0$, and each step of the nested computation can be computed as $R_{JKL\ldots} = [\nabla R_{KL\ldots}(\widetilde{\mathbf{\Lambda}}^\star)]_J \cdot \lambda_{JK}$ i.e., 'Gradient $\times$ Input (GI)'. This simple explanation procedure which we call GNN-GI will serve as a baseline in our experiments.

## 3 The GNN-LRP Method

The approach of Section 2.4 gives us a practical way of extracting higher-order explanations by analyzing interaction blocks individually. However, the analysis of each interaction block can itself be challenging. For example, the interaction block of a GIN can be composed of multiple layers. The high nonlinearity caused by these multiple layers can make it difficult if not impossible to choose a root point $\widetilde{\mathbf{\Lambda}}^\star$ at which a Taylor expansion accurately models the quantity to explain.

In the following, we consider an extension of the Taylor expansion method, called deep Taylor decomposition (DTD) [41]. It consists of replacing the Taylor expansion of the multilayer model by several Taylor expansions performed at each layer. When applied to standard neural networks, DTD yields an explanation technique called Layer-wise Relevance Propagation (LRP) [16], [42] that was shown to be more robust than simple gradient-based methods. Application of LRP to the multiple interaction blocks of a GNN yields a novel procedure for explaining GNNs that we call 'GNN-LRP'.

To simplify the exposition of our method, we will consider the special case of the graph convolutional network (GCN) [36]. In a GCN, each interaction block is composed of a linear aggregate function with positive adjacencies, followed by a linear/ReLU combine function. We restate these blocks by introducing the notation we use in this section

$$\begin{aligned} \text{aggregate:} \quad & z^a_K = \textstyle\sum_J \lambda_{JK} h^a_J \\ \text{combine:} \quad & h^b_K = \max\big(0, \textstyle\sum_a z^a_K w_{ab}\big). \end{aligned}$$

Here, $h^a_J$ denotes the activation of some neuron with index $a$ inside the node $J$. The notation $\sum_a$ represents a sum over all

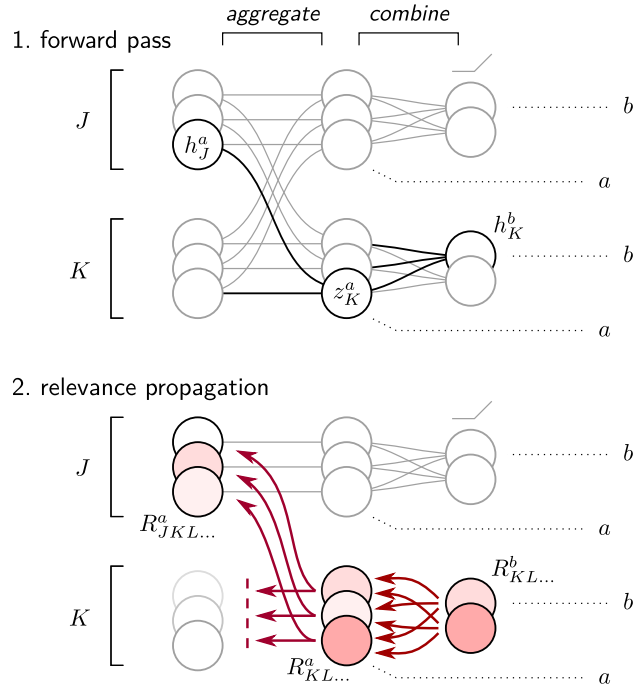


Fig. 3. Diagram of GNN-LRP annotated with variables that are used. (1) Sketch of an interaction block featuring an aggregation layer followed by a combine layer. The sketch shows nodes $J$ and $K$, each of them represented by neurons. Two neurons of consecutive layers of the combine function are denoted by $a$ and $b$. (2) GNN-LRP propagation flow, where we observe that only relevance scores that propagate along a particular walk in the input graph are retained.

neurons $a$ composing a node plus a hardcoded neuron '0' with activation $z_K^0 = 1$ and with $w_{0b}$ representing the bias. To further simplify the notation, we also omit the star symbol on the variable $\lambda_{JK}$. The aggregate and combine steps of a GCN are illustrated in Fig. 3 (top).

### 3.1 Deep Taylor Decomposition

Like in Section 2.4 we consider the problem of attributing $R_{KL\ldots}$ to the adjacencies $\lambda_{JK}$. Unlike Eq. (6), deep Taylor decomposition (DTD) adds further granularity to the attribution process, by considering relevance scores not at the node level, but at the neuron level (i.e., $R^b_{KL\ldots}$). Also, instead of decomposing this quantity directly to the adjacencies $\boldsymbol{\lambda}_K$, DTD considers as a first step a redistribution on the intermediate representation $\boldsymbol{z}_K$.

For this, DTD defines a 'relevance model', which is chosen here to be $\widehat{R}^b_{KL\ldots}(\boldsymbol{z}_K) = h^b_K(\boldsymbol{z}_K)\, c^b_{KL\ldots}$, a product of the neuron activation (which is a function of the intermediate representation), and a term that is constant and set in a way that the relevance model matches the true relevance $R^b_{KL\ldots}$ locally. (A justification for this relevance model is given in Section 3.2.) Using this relevance model, we can now attribute the relevance score to neurons of the intermediate representation by means of a first-order Taylor expansion at some root point $\widetilde{\boldsymbol{z}}_K$

$$R^{a\leftarrow b}_{KL\ldots} = \frac{\partial \widehat{R}^b_{KL\ldots}}{\partial z^a_K}\bigg|_{\widetilde{\boldsymbol{z}}_K} \cdot (z^a_K - \widetilde{z}^a_K). \tag{7}$$

(Note that the root point $\widetilde{\boldsymbol{z}}_K$ will typically be different for each output neuron $b$.) An aggregate relevance score is then obtained by summing contributions from neurons in the layer above, i.e., $R^a_{KL\ldots} = \sum_b R^{a\leftarrow b}_{KL\ldots}$. The next step is to attribute the newly computed relevance scores to the adjacencies $\lambda_{JK}$ in the aggregate step. For this, we proceed similarly to above, by first defining a relevance model $\widehat{R}^a_{KL\ldots} = z^a_K(\boldsymbol{\lambda}_K)\, c^a_{KL\ldots}$ and then computing the first-order terms of a Taylor expansion

$$R^a_{JKL\ldots} = \frac{\partial \widehat{R}^a_{KL\ldots}}{\partial \lambda_{JK}}\bigg|_{\widetilde{\boldsymbol{\lambda}}_K} \cdot (\lambda_{JK} - \widetilde{\lambda}_{JK}). \tag{8}$$

The overall layered attribution procedure is illustrated in Fig. 3 (bottom).

### 3.2 Deriving GNN-LRP Propagation Rules

The equations above define a general framework for layer-wise propagation of the relevance to the adjacencies $\lambda_{JK}$. To arrive at concrete propagation rules, it remains to set the root points in Eqs. (7) and (8). For the first Taylor expansion, we choose $\widetilde{\boldsymbol{z}}_K$ at the intersection of the line

$$\{\boldsymbol{z}_K - s\boldsymbol{z}_K \odot (\mathbf{1} + \gamma \mathbf{1}_{\boldsymbol{w}_b \geq \mathbf{0}}) \mid s \in \mathbb{R}\},$$

and the ReLU hinge. The parameter $\gamma$ tilts the search for a root point towards neurons with positive contributions. A high value of $\gamma$ leads to root points $\widetilde{\boldsymbol{z}}_K$ that are closer to the activation $\boldsymbol{z}_K$ and that better contextualize the explanation. Injecting this root point in Eq. (7) gives the relevance messages $R^{a\leftarrow b}_{KL\ldots} = z^a_K\, w_{ab}\, s\, (1 + \gamma 1_{w_{ab} \geq 0})\, c^b_{KL\ldots}$. Resolving the parameter $s$ and pooling relevance messages coming from the multiple output neurons, we obtain the propagation rule

$$R^a_{KL\ldots} = \sum_b \frac{z^a_K(w_{ab} + \gamma w^+_{ab})}{\sum_a z^a_K(w_{ab} + \gamma w^+_{ab})} R^b_{KL\ldots}. \tag{9}$$

For propagation in the aggregate layer, applying a similar root search strategy as above yields the root point $\widetilde{\boldsymbol{\lambda}}_k = 0$. Injecting this root point in Eq. (8) gives the propagation rule

$$R^a_{JKL\ldots} = \frac{\lambda_{JK} h^a_J}{\sum_J \lambda_{JK} h^a_J} R^a_{KL\ldots}. \tag{10}$$

We are now in position to verify the validity of the relevance model we have used in Section 3.1 to perform the Taylor expansions. An inspection of Eqs. (9) and (10) shows that the relevance score, resulting from applying these rules, can always be written as a product of the corresponding activation and a term that depends on this activation only through two nested sums. This provides a justification for our relevance model which approximates the latter term as constant. (The same argument can be found in [41] for standard deep neural networks).

Interestingly, the rules in Eqs. (9) and (10) can be merged into a single propagation rule that we show in Eq. (11) of Table 2. The propagation rule can be seen as a generalization of the LRP-$\gamma$ rule [42] to the GCN. It inherits some of its theoretical properties such as the connection between LRP and Gradient × Input (GI). In particular, in the limit of $\gamma \to 0$, the explanations produced by GNN-LRP become equivalent to those of the GNN-GI baseline. Our experiments will however demonstrate that higher values of $\gamma$ produce better explanations.

TABLE 2
Practical GNN-LRP Propagation Rules for Different Types of GNNs

| Model | Aggregate | Combine | GNN-LRP Rule |
|---|---|---|---|
| GCN [36] | $\boldsymbol{Z}_t = \boldsymbol{\Lambda}\boldsymbol{H}_{t-1}$ | $\boldsymbol{H}_t = \rho(\boldsymbol{Z}_t\boldsymbol{W}_t)$ | $R^a_{JKL\dots} = \sum_b \frac{\lambda_{JK} h^a_J w^{\uparrow}_{ab}}{\sum_{J,a} \lambda_{JK} h^a_J w^{\uparrow}_{ab}} R^b_{KL\dots}$ (11) |
| GIN [44] | $\boldsymbol{Z}_t = \boldsymbol{\Lambda}\boldsymbol{H}_{t-1}$ | $\boldsymbol{H}_t = (\mathrm{MLP}^{(t)}(\boldsymbol{Z}_{t,K}))_K$ | $R^a_{JKL\dots} = \sum_b \frac{\lambda_{JK} h^a_J}{\sum_J \lambda_{JK} h^a_J} \mathrm{LRP}(R^b_{KL\dots}, z^a_K)$ (12) |
| Spectral [43], [45] (case $\lambda \geq 0$) | $\boldsymbol{Z}_{s,t} = \boldsymbol{\Lambda}_s \boldsymbol{H}_{t-1}$ | $\boldsymbol{H}_t = \rho(\sum_s \boldsymbol{Z}_{s,t}\boldsymbol{W}_{s,t})$ | $R^a_{JKL\dots} = \sum_b \frac{\sum_s \lambda^s_{JK} h^a_J w^{s\uparrow}_{ab}}{\sum_{J,a} \sum_s \lambda^s_{JK} h^a_J w^{s\uparrow}_{ab}} R^b_{KL\dots}$ (13) |

*The function $\rho(\cdot)$ is the rectification function $\max(0,\cdot)$. We denote by $w_{ab}$ the element of the matrix $\boldsymbol{W}_t$ that links neuron $a$ to neuron $b$, and we use the notation $w^{\uparrow} = w + \gamma w^{+}$. In the last row, $\boldsymbol{\Lambda}_s$ represents one component of the graph convolutional filter approximation presented in [43].*

### 3.3 Application of GNN-LRP Beyond the GCN Model

The derivation above has focused on the simple case of the GCN model. However, the procedure can be extended to a broad range of other models. For example, the *GIN* [44] can be seen as an extension of the GCN where the combine function consists of multiple layers. Here, we simply need to apply LRP rules in each layer of the combine function followed by the aggregation layer (Eq. (12)). The procedure can again be justified as a deep Taylor decomposition. Spectral filtering methods [46] such as the *Spectral Network* [43] or *ChebNet* [45] can be viewed as variants of the GCN with multiple adjacency matrices, and GNN-LRP propagation rules can also be derived for these models (Eq. (13)).

GNN-LRP can be applied to further GNN models such as the original GNN model [1], *GraphSAGE* with mean aggregation [47], or *Neural FP* [48]. GNN-LRP is also applicable to other recent GNN architectures such as the *SchNet* [37] used for predicting molecular properties, and where the graph is a representation of the distance between atoms. GNN-LRP is also applicable to convolutional neural networks for computer vision such as *VGG-16* [49], which can be seen as a particular GNN receiving as input a pixel lattice. GNN-LRP can be extended to more advanced architectures such as joint CNN-GNN models for spatio-temporal graphs [38]. Furthermore, GNN-LRP allows in principle to use different edges and nodes at different layers, which can be useful to handle graph pooling structures, such as those described in [50].

More generally, GNN-LRP procedures can be designed for any architecture that is expressible as an alternation of aggregate and combine steps as given in Eqs. (1) and (2). The combine step is then treated as a common neural network, built on some lower layer of activations. Here, existing LRP rules, that have been developed for a variety of neural network models (e.g., [16], [42], [51]) can be applied. In the aggregation step we either have a linear pooling over nodes such as mean or sum aggregation, or a nonlinear pooling, such as max-pooling in GraphSAGE [47]. In the linear case, the same propagation rule as for the GCN in Eq. (10) can be applied. In the nonlinear case, one needs to build specific propagation rules, e.g., using the deep Taylor decomposition approach of Section 3.1.

### 3.4 Implementing GNN-LRP

As an extension of LRP, GNN-LRP inherits several tricks that strongly facilitate the implementation of the method compared to a direct transcription of Eqs. (11), (12), and (13) into code. One such trick is the use of forward/backward hooks to alter the gradient computation in a way that it matches the LRP signal.

For example, GNN-LRP for a GCN can be easily implemented by rewriting the combine function of each interaction block as

$$\begin{aligned}
\boldsymbol{P}_t &\leftarrow \boldsymbol{Z}_t \boldsymbol{W}_t^{\uparrow} \\
\boldsymbol{Q}_t &\leftarrow \boldsymbol{P}_t \odot [\rho(\boldsymbol{Z}_t\boldsymbol{W}_t) \oslash \boldsymbol{P}_t]_{\mathrm{cst.}} \\
\boldsymbol{H}_t &\leftarrow \boldsymbol{Q}_t \odot \boldsymbol{M}_K + [\boldsymbol{Q}_t]_{\mathrm{cst.}} \odot (\boldsymbol{1} - \boldsymbol{M}_K),
\end{aligned}$$

where $[\cdot]_{\mathrm{cst.}}$ detaches the quantity to which it applies from the gradient and $\boldsymbol{M}_K$ is a mask that retains node $K$. More details are given in Appendix C in the Supplement, available online. Along with automatic differentiation capabilities of neural network software and the availability of predefined layers such as convolution or pooling, this implementation trick allows to implement GNN-LRP for complex GNN architectures without much code overhead. This implementation trick is also used in a GNN-LRP demo code that we provide at https://git.tu-berlin.de/thomas_schnake/demo_gnn_lrp.

### 3.5 Limitations

When implementing the masking approach from above, a limitation of GNN-LRP is the need to compute in the general case as many forward-backward passes as there are walks in the input graph. This typically limits the applicability of the method to GNNs of a limited depth (e.g., three or four layers). Ability of GNN-LRP to scale to bigger graphs and deeper models is tied to the application requirements, e.g., whether coarse-graining of certain nodes is permitted so that multiple relevant walks can be merged into a single computation, or whether only the most relevant walks are of interest, in which case pruning techniques can be applied.

Some explanation techniques such as GNNExplainer [26], PGExplainer [27] and SubgraphX [32] are based solely on evaluating the function or its gradient multiple times. This enables an application of the method without further knowledge of the model implementation. This is not the case for GNN-LRP (and other explanation methods such as GraphMask [30]), which require usage of the neural network internals. Our GNN-LRP method, in particular, requires access to the representation at each layer in order to implement appropriate propagation rules at each layer.

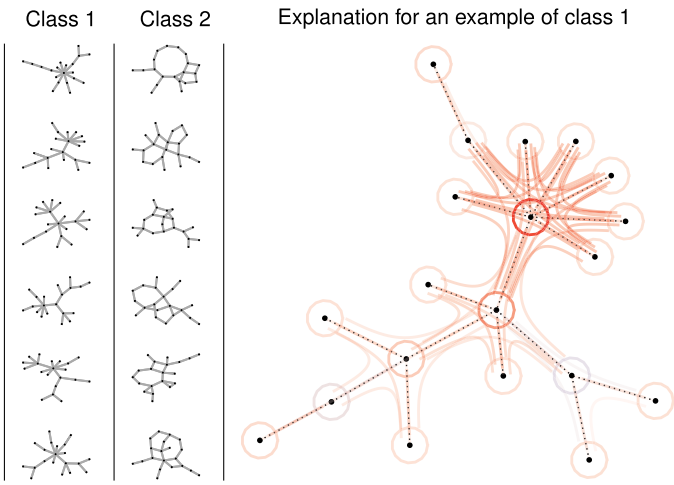


Fig. 4. *Left:* Examples from the two classes of the BA-growth dataset. *Right:* GNN-LRP explanation for the GIN prediction of a graph of class 1. Relevant (positively contributing) walks are shown in red and negatively contributing walks are in blue. Circles represent walks (or part of the walks) that are stationary.

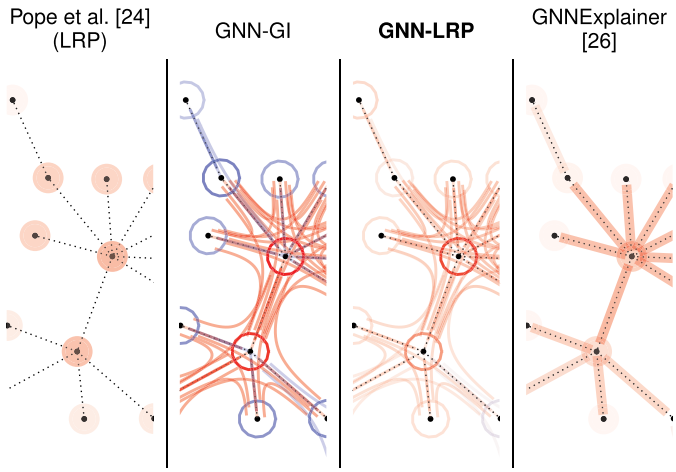


Fig. 5. Comparison of different explanation techniques on the same graph as in Fig. 4. GNN-LRP produces more detailed explanations compared to [24] and [26], and brings more robustness compared to GNN-GI.

This incurs an implementation overhead, which remains however manageable if making use of the implementation trick mentioned in the section above.

## 4 Evaluation of GNN-LRP

To test the proposed GNN-LRP method, we train various types of GNNs on several graph prediction tasks. We start with a two-class synthetic problem that we call *BA-growth* where the first class consists of Barabási-Albert graphs [52] of growth parameter 1 (class 1), and where the second class has a slightly higher growth model and new nodes are attached preferably to low-degree nodes (class 2). Examples of graphs from the two classes are given in Fig. 4 (left). Details on this synthetic dataset are given in Appendix E in the Supplement, available online.

We consider a graph isomorphism network (GIN) that we train on this task. Our GIN has two interaction blocks. In each interaction block, the 'combine' function consists of a two-layer network with 32 neurons per node at each layer. The initial state $\boldsymbol{H}_0$ is an all-ones matrix of size $n \times 1$, i.e., nodes do not have intrinsic information. The GIN receives as input the connectivity matrix $\boldsymbol{\Lambda} = \widetilde{\boldsymbol{A}}/2$ where $\widetilde{\boldsymbol{A}}$ is the adjacency matrix augmented with self-connections. The GIN is trained on this task until convergence, where it reaches an accuracy above 95 %. More details on the model and its training are given in Appendix F.1 of the Supplement, available online. After training, we take an exemplary input graph from class 1, predict it with our GIN, and apply GNN-LRP on the prediction. We use the LRP parameter $\gamma = 2$ and $\gamma = 1$ in each layer of the first and second interaction blocks respectively. The resulting explanation is shown in Fig. 4 (right).

The explanation produced by GNN-LRP reveals that walks that traverse or stay in the high-degree node are the principal contributors to the GIN prediction. On the other hand leaf nodes or sequences of low-degree nodes are found to be either irrelevant or to be in slight contradiction with the prediction.

In the following, we compare GNN-LRP to a selection of other GNN explanation methods:

(a) *Pope et al.* [24]: The method views the GNN as a function of the initial state $\boldsymbol{H}_0$, and performs an attribution of the GNN output on nodes as represented in $\boldsymbol{H}_0$. In principle, the proposed framework lets the user choose the technique to perform attribution on $\boldsymbol{H}_0$. In our benchmark, we use the techniques Gradient × Input (GI) and LRP.

(b) *GNN-GI*: This simple baseline replaces in the GNN-LRP procedure the LRP steps by Gradient × Input steps. It can also be seen as a special case of GNN-LRP with parameter $\gamma = 0$.

(c) *GNNExplainer* [26]: The method runs an optimization problem that finds a selection of edges that maximizes the model output. The procedure can be viewed as finding a mask $M = \sigma(\boldsymbol{R})$ where $\sigma$ denotes the logistic sigmoid function, that maximizes the prediction $f(M \odot \boldsymbol{\Lambda})$. The explanation is then given by $\boldsymbol{R}$.

Explanations produced by each method are shown in Fig. 5. The method by Pope *et al.* [24] highlights nodes that are relevant for the prediction. However, it is difficult to determine from the explanation whether the highlighted nodes are relevant by themselves or if they are relevant in relation to their neighbors. The GNN-GI baseline we have contributed, and our more advanced method GNN-LRP, provide a much higher level of granularity, distinguishing between the contribution of the node and its interaction with other nodes. In comparison to GNN-LRP, however, the GNN-GI baseline tends to be less selective, with spurious positive or negative relevance, and generally more noisy (something we will also observe later in Section 5.3). The GNNExplainer [26] produces explanations that are in agreement with GNN-LRP but less detailed.

Overall, from our first qualitative inspection, GNN-LRP is the only method in our benchmark that produces explanations that have both the desired robustness and a high level of detail.

### 4.1 From Attribution to Subgraph Selection

We would like to compare the methods above *quantitatively*. A first difficulty is that most quantitative benchmarks are

not making direct use of attribution scores, nor do they operate at a specific granularity such as edges or walks. Instead, two common evaluation methods that we will consider in Sections 4.2 and 4.3 require to output a maximally relevant subgraph or sequence of subgraphs.

We introduce a technique to extract most relevant subgraphs from relevance scores obtained by an attribution method. Our approach is applicable equally to node-based, edge-based, and walks-based explanations and it makes efficient use of the higher-order information contained in the latter explanations to extract more precise subgraphs.

Let $\mathcal{X}$ denote our unit of attribution. For a node-based attribution (e.g., [24]), an edge-based attribution (e.g., GNNExplainer [26]), or walks-based attribution, we replace $\mathcal{X}$ by a node $\mathcal{N}$, edge $\mathcal{E}$ or bag-of-edges $\mathcal{B}$ respectively. We define the relevance of a subgraph $\mathcal{S} \subset \mathcal{G}$ to be the relevance of subfeatures the subgraph is composed of. This means that a given subgraph $\mathcal{S}$ is assigned the score

$$R_{\mathcal{S}} = \sum_{\mathcal{X} \in \mathcal{S}} R_{\mathcal{X}}. \tag{14}$$

The best subgraph is then ideally chosen via the optimization problem

$$\mathcal{S}^{*} = \arg\max_{\mathcal{S} \in \mathbb{S}} R_{\mathcal{S}}, \tag{15}$$

where $\mathbb{S}$ is the set of admissible subgraphs, e.g., all graphs composed of a given number of nodes. In the worst case, we have exponential complexity to obtain $\mathcal{S}^{*}$, which is especially costly for bag-of-edges attributions. Approximation schemes that we use (*random sampling*, *local best guess*) for these optimization problems are given in Appendix D in the Supplement, available online.

### 4.2 Model Activation Task

Our first evaluation experiment is called 'model activation' and is inspired by pixel-flipping[1] [53]. We start with an empty subgraph $\mathcal{S}$ and grow it by adding nodes one-by-one. Nodes are selected at each step to incur a maximum growth of the relevance score $R_{\mathcal{S}}$. While nodes are being added to the graph, we keep track of the true GNN output $f(\mathcal{S})$. The higher the GNN output, the more faithful the explanation technique, as the latter was therefore able to identify the correct substructure. Pseudo-code for our activation task is given in Algorithm 1.

The algorithm returns an *area under the activation curve* (AUAC) score. The higher the AUAC score the better the explanation technique. The procedure is repeated for a sufficiently large number of data points, leading to an averaged AUAC score.

We now use the AUAC metric to evaluate the different explanation methods on a broad set of architectures and datasets. On the *BA-growth* dataset, we train a GCN, a GIN, and a spectral network, each of them with two interaction blocks, and with respectively 128, 32 and 32 neurons per node at each layer. For the spectral network, we give in each layer the power expansion $\mathbf{\Lambda} = [\widetilde{\boldsymbol{A}}^0, \frac{1}{2}\widetilde{\boldsymbol{A}}^1, \frac{1}{4}\widetilde{\boldsymbol{A}}^2]$ as input. All models perform almost perfectly on the classification task. More details on the models and their training can be found in Appendix F.1 of the Supplement, available online.

1. Pixel-flipping [53] starts with an original data point and 'flips' (i.e., destroys) input features from most to least relevant according to the explanation, and how quickly the output of the model drops throughout the flipping process. The faster the output drops, the more faithful the explanation.

TABLE 3
AUAC Scores of Each Explanation Method on Various Datasets and Models

| | P [24] (GI) | P [24] (LRP) | GNN-GI (ours) | GNN-LRP (ours) | GNNExpl [26] | Random |
|---|---|---|---|---|---|---|
| BA-growth, GCN | 2.54 | 3.02 | 2.93 | **3.52** | 3.32 | 1.05 |
| BA-growth, GIN | 2.75 | 3.49 | 3.04 | **3.84** | 3.71 | 1.18 |
| BA-growth, spectral | 0.29 | **1.91** | 0.23 | 1.85 | 1.65 | 0.09 |
| SST, GCN | 26.07 | 26.35 | 26.40 | 26.65 | **27.07** | 20.47 |
| SchNet-$E$ [37] | 10.39 | | **10.47** | | 10.41 | 8.00 |
| SchNet-$\mu$ [37] | 0.87 | | **1.09** | | 1.01 | 0.38 |
| VGG-16 [49] | 9.46 | 13.18 | 12.03 | **14.04** | — | 7.90 |

*All values are averages over 200 data points. The higher the score the better the explanation. Best performers are shown in bold. The results on SchNet-E are scaled by a factor of* $10^{-3}$.

**Algorithm 1.** Computation of the AUAC Metric

**Input:** Ordered sequence of nodes $\mathcal{N}^{(1)}, \ldots, \mathcal{N}^{(|\mathcal{G}|)}$ for which $\sum_{i=1}^{|\mathcal{G}|} R_{\{\mathcal{N}^{(1)}, \ldots, \mathcal{N}^{(i)}\}}$ is maximized.

```
AC ← []
S ← ∅
for i = 1...|G| do
    S ← S ∪ {N^(i)}                ▷ Add node to graph
    AC.append(f(S))
end for
AUAC ← mean(AC)
return AUAC
```

In addition, we also consider networks trained on real data: A first one is designed and trained by ourselves on the Stanford Sentiment Treebank (*SST*) [54] dataset, where the graph represents a syntactic tree with nodes carrying information about each word. Another one is the *SchNet* [37] model used for molecular prediction where nodes and edges represent atom types and distances respectively. Finally, we use a pre-trained *VGG-16* [49] network for image recognition that we interpret as a graph neural network operating on a lattice of size $14 \times 14$ and starting at convolutional block 3. In this last network, each node represents the collection of activations at a specific spatial location. Details for each network are given in Appendix F in the Supplement, available online. Results for the activation task for each network and explanation method are summarized in Table 3.

On the *BA-growth* dataset, we observe that GNN-LRP outperforms other methods on average. Nearest competitors are Pope *et al.* [24] (in combination with LRP), and the GNNExplainer [26]. This result corroborates our qualitative analysis at the beginning of Section 4. We would also like to verify to which extent GNN-LRP is sensitive to its

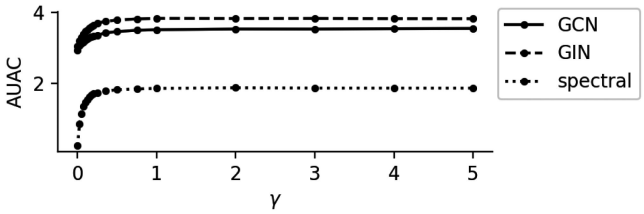


Fig. 6. The effect of the parameter $\gamma$ of GNN-LRP on the AUAC score on the BA-growth dataset. Higher values of $\gamma$ give better performance.

hyperparameters [55]. The sensitivity of GNN-LRP towards its hyperparameter $\gamma$ is shown in Fig. 6. We observe that any choice of parameter $\gamma \geq 1$ delivers roughly the same high benchmark performance. The lowest performance is at $\gamma = 0$ which is equal to attributing with GNN-GI.

On the SST task, GNNExplainer and GNN-LRP [26] come first and second, followed by GNN-GI and both node attribution methods. In natural language, the sentiment associated to a sentence relies on word combinations, e.g., negation. This can explain why methods which attribute on interactions of words such as GNN-LRP and GNNExplainer perform better.

For the experiments on the *SchNet* model, our method performs again above competitors. Note that in this experiment, we find that using the LRP parameter $\gamma = 0$, which is equivalent to Gradient $\times$ Input (cf. [56], [57]), already gives good explanations.[2] Hence, GNN-GI and GNN-LRP have the same performance in this case. The difference to other competitors (Pope *et al.* [24] and GNNExplainer [26]) is small on the prediction of the energy $E$ but larger for the dipole-moment $\mu$, possibly because of a more complex structure of the prediction task involving longer interactions.

Finally, on the *VGG-16* image recognition model, GNN-LRP again produces the highest AUAC score. Here, the superiority of LRP over GI can be explained both by a better handling of neuron biases and by a higher robustness to shattered gradients, a key difficulty to account for when explaining very deep models [58], [59].

In a similar experiment to the activation task we search for the features that least effect the model output when removing them from the input graph. We refer to this validation method as the 'pruning task' in which we compute the area under the pruning curve (AUPC). We found that GNN-LRP outperforms most of the other baseline methods in each of the experimental setups as well. For more details on the pruning task and its results, we refer to Appendix G in the Supplement, available online.

We also found that the activation and pruning tasks are very similar to the *fidelity* and *sparsity* tasks presented in [24] and [35]. We saw that a good performance for the AUAC and AUPC aligns with a good performance in the fidelity metric. In addition we point out that the activation and pruning tasks also reflect if the explanation method differentiate properly between important and redundant graph features, which is similar to what the *sparsity* measure does. We conclude that the activation and pruning tasks already reflect a variety of existing validation methods. For more details on the comparability of the evaluation metrics with additional quantitative experiments, we refer to Appendix H in the Supplement, available online.

Overall, we find that GNN-LRP is systematically the best method in our benchmark. With the fine-grained yet robust explanations it provides, GNN-LRP is capable of precisely and contextually identifying elements of the graph that contribute the most or the least to the prediction.

### 4.3 BA-2motifs Benchmark

As a second quantitative evaluation, we consider the *BA-2motifs* [27] benchmark that comes with 'ground-truth' explanations. In this dataset, the class of each data point can be traced to a particular motif in the input graph. We stress that, in contrast to the model activation task from Section 4.2, this benchmark, to detect the motifs in each data point, is only meaningful if the model solves the problem well. Indeed, only if the model exhibits a good performance without fraudulent Clever Hans strategies, we can expect that the benchmark measures the true quality of the attribution method.

For the benchmark evaluation we use the GIN architecture similar to the GIN introduced at the beginning of Section 4, which predicts the BA-2motifs dataset almost perfectly, with an accuracy of 99.9%. For the exact model architecture and its training we refer to Appendix F.1.2 of the Supplement, available online. We also note that when using GNN-LRP in this section we select the hyperparameter $\gamma = 3$, $\gamma = 1.5$ and $\gamma = 0$ in the first, second and third interaction block respectively. The BA-2motifs benchmark assumes that explanations are given as an ordered sequence of nodes (or edges) from most to least relevant. It then computes the area under the receiver operating characteristic (AUROC) of this ordered sequence against the ground-truth subgraph. To produce a good sequence of nodes/edges, we use the approach described in Section 4.1, starting with an empty graph and adding the nodes/edges in a way that that the resulting sequence of subgraphs has its relevance scores summing to the largest possible value. When the optimal subgraph sequence is too expensive to compute, we use an approximation scheme, namely the we use a *random sampling approach*, which we discuss in more detail in Appendix D of the Supplement, available online. In the random sampling approach we choose the hyperparameter $k$, which represents the number of feature orderings generated randomly, to be 100 and 150 when considering a sequence of ordered nodes and ordered edges respectively. For the special case where we need to produce a sequence of edges from a node-based attribution (as in [24]), we generate the edge scores by summing the relevance scores of the containing nodes.

In Table 4 we see the AUROC for different attribution methods divided into the cases where the explanation is given as a sequence of nodes or edges (these scenarios are referred to as 'node classification' and 'edge classification'). We see that for the node case, explanation methods Pope *et al.* (LRP), GNN-LRP, PGExplainer and GNNExplainer, all have an AUROC score above 0.9, which shows that all these explanation methods have been able to extract from the model the class-specific motif in the input graphs. The GI methods (namely P [24] (GI) and GNN-GI) perform poorly for the node and edge classification, and are comparable to random

2. This can be explained by the presence in SchNet of residual connections that reduce gradient noise, and also by the fact that some of the nonlinearity of SchNet occurs in the mapping of distance between atoms onto basis functions, which our explanation technique views as constant and does not propagate through.

TABLE 4
AUROC of Selected Explanation Methods on BA-2motifs

| | P [24] (GI) | P [24] (LRP) | GNN-GI (ours) | GNN-LRP (ours) | GNNExpl [26] | PGExpl [27] |
|---|---|---|---|---|---|---|
| AUROC, nodes | 0.47 | 0.97 | 0.5 | **1.0** | 0.94 | × |
| AUROC, edges | 0.45 | 0.95 | 0.47 | **0.98** | 0.81 | *0.93 ±0.02* |

*The AUROC computation is differentiated between node and edge classification. All values are averages over 200 data points. The values for PGExplainer are extracted from [27].*

feature ordering. The method with the best performance for both feature qualities, is GNN-LRP with a perfect performance in the case of nodes and an almost perfect score by 0.02 in the case of edges. This is an improvement of 0.05 to the previous baseline performance in [27]. We stress, that we only used approximation schemes for finding the subgraph ordering, hence we can potentially expect further performance gains if using more advanced algorithms for optimizing subgraph sequences.

Overall, we have shown that among all methods we have tested, GNN-LRP performs best in finding motifs in the input graph that give rise to the graph's correct classification.

### 4.4 Sanity Checks and Other Evaluations

As an additional evaluation, we perform the *sanity checks* proposed in [60], which test if the explanation methods are sensitive towards model randomization. We randomized the parameter of a GCN model composed of two interaction blocks, and trained on the BA-growth dataset, as described in the beginning of Section 4. We found that the randomization of any layer has a profound effect on the heatmaps of GNN-LRP and GNN-GI. In addition, for all interpretation methods the relevance scores of the original and randomized models deviate drastically. We therefore conclude that all interpretation methods pass the sanity test. For more details on the randomization tests we refer to Appendix I of the Supplement, available online.

Another work [61] introduces further metrics for evaluating graph neural networks, called *accuracy*, *consistency*, *faithfulness*, and *stability*. Faithfulness and accuracy can be related to our sanity checks and the analysis in Section 4.3 respectively. Consistency makes the assumption that each high-performing model has a similar strategy, which does not account for potential Clever Hans cases (we show Clever-Hans-type strategies in the application sections). Finally, attribution stability verifies that the explanation remains the same under a small perturbation of the input. Here, we note that our proposed GNN-LRP method inherits some properties of LRP, in particular, it is not subject to potential discontinuities of the model's gradient [17]. It therefore has better built-in stability properties compared to gradient-based methods such as GI or GNN-GI.

As an additional remark, we would like to point out that GNN-LRP—like for any explanation method in our benchmark—is not developed with built-in robustness to adversarial attacks [62]. Hence, we recommend to restrict its usage to non-adversarial scenarios.

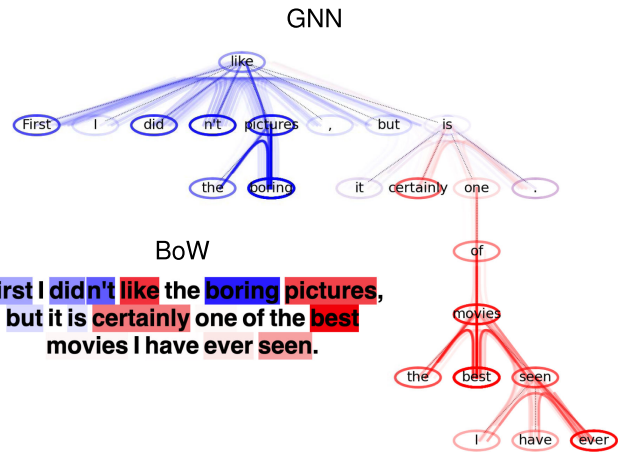


Fig. 7. Sentence predicted by the GNN and the BoW model, and explained by GNN-LRP (applied on the difference between the positive and negative sentiment logit, and with the LRP parameter $\gamma = 3$). Contributions to positive sentiment are in red, and contributions to negative sentiment are in blue.

## 5 New Insights With GNN-LRP

Having validated the proposed GNN-LRP method on a diverse set of GNNs including state-of-the-art models, and having shown the multiple advantages of our method compared to previous approaches, we will now inspect the explanations produced by GNN-LRP on some of these practically relevant GNN models to demonstrate how useful insights can be extracted about the GNN model and the task it predicts.

### 5.1 Sentiment Analysis

In natural language processing (NLP) text data can be processed either as a sequence, or with its corresponding grammatical structure represented by a parse tree [63], [64]. The latter serves as an additional structural input for the learning algorithm, to incorporate dependencies between words. NLP tasks are therefore particularly amenable to GNNs since these models can naturally incorporate the graph structure.

In the following experiments, we will demonstrate how GNN-LRP can be used to intuitively and systematically assess the quality of a GNN model, including its overall prediction strength and also its few weaknesses. For this, we will consider a GCN composed of two interaction layers, to classify sentiments in natural language text [65]. We train our model on the Stanford Sentiment Treebank (SST) [54].[3] (Note that this example serves as a mere demonstration for the versatility of our explanation approach and is by no means intended to reflect or compete with state-of-the-art NLP systems.) For details on the experimental setup we refer the reader to Appendix F.2 in the Supplement, available online.

In Fig. 7 (top) we show an example of a GNN-LRP explanation for some exemplary input sentence containing a mixture of positive and negative sentiment. We observe that distinct combinations of words give rise to the emphasis of these sentiments. The model correctly detects word combinations such as "*the best movies*" to carry a positive sentiment, and "*boring pictures*" to contribute negatively.

3. https://nlp.stanford.edu/sentiment/index.html

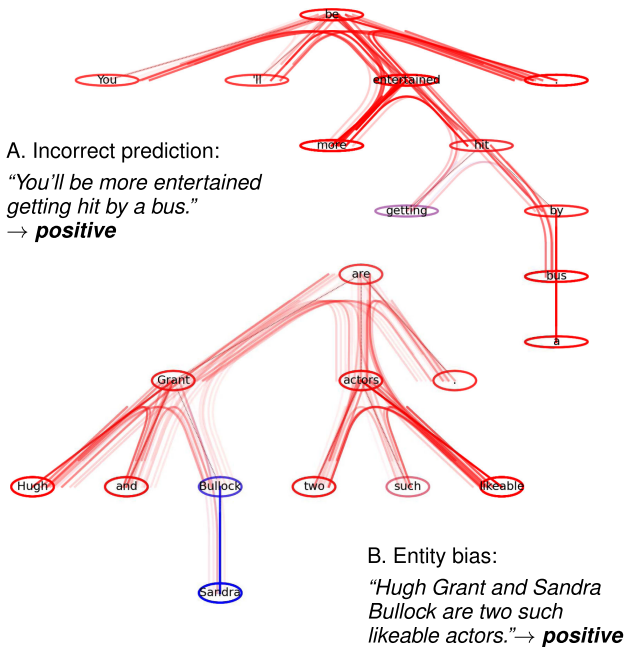


Fig. 8. Two selected examples of dependency trees from the SST dataset, predicted by the GNN model to be positive, and for which GNN-LRP highlights a flaw in the prediction strategy.

GNN-LRP can also be used to assess the GNN prediction strategy relative to other (simpler) models, such as Bag-of-Words (BoW). The BoW model can be seen as a GNN model with zero interaction layers, hence our explanation technique applies to that model as well. Fig. 7 (bottom) shows the explanation of the BoW prediction for the same sentence as for the GNN. Interestingly, several words are now attributed a sentiment different from the one obtained with the GNN. For example, "*like*" becomes positive, which however appears in contradiction with the preceding words "*didn't*". Hence, GNN-LRP has highlighted from a single sentence that the GNN model is able to properly capture and disambiguate the sentiment of consecutive words, whereas the BoW model is not.

In the next experiment, we consider two additional sentences from the SST corpus, where GNN-LRP contributes to uncovering or better understanding flaws of a trained GNN model. Fig. 8 A shows a data sample that contains sarcasm and that is falsely classified by the GNN to be positive. Our explanation method highlights that relevant walks are too localized to capture the interaction between words that jointly explain sarcasm. Instead, local positive interactions such as "*more entertained*" dominate the prediction, which leads to the incorrect prediction. In Fig. 8 B, we see a case of *entity bias* where the GNN model is biased towards particular entities, namely "*Hugh Grant*" and "*Sandra Bullock*". In this example, GNN-LRP finds that the GNN model uses with no objective reason "*Hugh Grant*" and "*Sandra Bullock*" as evidence for positive and negative sentiment, respectively. In NLP model biases are well studied areas and some approaches to tackle that problem have already been developed [66], [67].

To identify words or combinations of words that systematically contribute to a positive or negative sentiment—and potentially discover further cases of entity bias,—we apply GNN-LRP on the whole dataset. This lets us find the combination of words (given by a walk $\mathcal{W}$) that are on average the most positive/negative (according to their score $R_{\mathcal{W}}$). Fig. 9 shows top-3 walks of both types (positive and negative) and containing one to three unique words.

| | | |
|---|---|---|
| *solidly* | *documentary brilliant* | *'s [is] astonishing .* |
| *witty* | *brilliant !* | *brilliant ! '* |
| *lewis* | *solidly seaworthy* | *resourceful and ingenious* |
| ⋮ | ⋮ | ⋮ |
| *horrible* | *sorry charlie* | *bears bad is* |
| *boring* | *no more* | *sorry , charlie* |
| *ridiculous* | *ridiculous .* | *no more .* |

Fig. 9. Walks that contribute the most to positive and negative sentiment according to the GNN, split by the number of unique words they contain.

We see that the walks which are most relevant for the task contain positive adjectives and adverbs, such as "*solidly*", "*brilliant*", "*witty*" or "*astonishing*". The walks with a very negative relevance score contain negative words such as "*ridiculous*", "*boring*" or "*horrible*", but also subsequences like "*sorry, charlie*" or "*no more*." which clearly transport a negative emotion. Here again, GNN-LRP detects an entity bias by the word "*lewis*", which the GNN model considers to be positively contributing although this word is objectively neutral. Note that this time, this entity bias was discovered directly, without having to visualize a large number of explanations.

Overall, applying the proposed GNN-LRP explanation method to the GNN model for sentiment classification has highlighted that GNN predictions are based on detecting meaningful sentence sub-structures, rather than single words as in the BoW model. Furthermore, GNN-LRP was able to find the reasons for incorrect predictions, or to shed light on potential model biases. The latter could be identified manually by visual inspection of many explanations, or systematically by averaging the GNN-LRP results on a whole corpus.

### 5.2 Quantum Chemistry

In the field of machine learning for quantum chemistry [68], [69], [70], [71], [72], GNNs have been exhibiting state of the art performance for predicting molecular properties [3], [37], [39], [73]. Such networks incorporate a graph structure of molecules either by the covalent bonds or the proximity of atoms.

In this section we will test the ability of GNN-LRP to extract meaningful domain knowledge from these state-of-the-art GNNs. We will consider for this the *SchNet*,[4] a GNN for the prediction of molecular properties [3], [37], [74], and we set the number of interaction blocks in this GNN to three. We train the model on the atomization energies and the dipole moments for 110,000 randomly selected molecules in the QM9 dataset [75].[5] For more details on the model parameters and the network architecture, we refer to Appendix F.3 in the Supplement, available online. On a test set of 13,885 molecules, the atomization energy and dipole moment are predicted well with a mean absolute error (MAE) of 0.015 eV and 0.039 Debye, respectively.

Our first objective is to get an insight into what structures in the molecule contribute positively or negatively to the

4. https://github.com/atomistic-machine-learning/schnetpack
5. https://doi.org/10.6084/m9.figshare.c.978904.v5

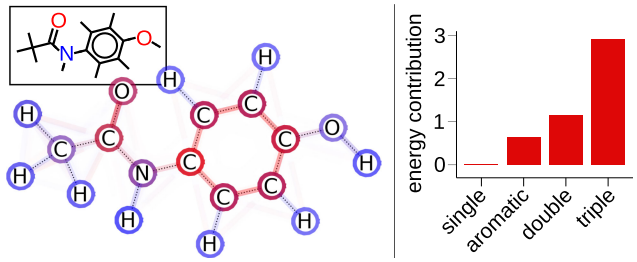


Fig. 10. *Left:* Paracetamol molecule, and the GNN-LRP explanation of its predicted energy. Red and blue indicate positive and negative contributions. Opacity indicates the magnitude of these contributions. *Right:* Average energy contribution per bond, depicted for each bond type separately, in arbitrary units.

molecule's energy. While it is common to look at the atomization energy (describing the energy difference to dissociated atoms), we consider here for the purpose of explanation the centered negative atomization energy, and we define this quantity to be the actual 'energy'. With this definition, molecules have high energy when they are hard to break and typically formed of strong bonds, and conversely, molecules have low energy when they are easy to break and unstable.

We consider for illustration the case of the *paracetamol* (acetaminophen) molecule, and feed this molecule to SchNet. Once the SchNet model has predicted its energy, we apply the GNN-LRP analysis in order to produce an explanation of the prediction. The resulting explanation is shown in Fig. 10 (left). We observe that the explanation is dominated by self-walks (i.e., staying in a single atom) or one-edge walks (traversing a single edge of the graph, in most cases, a bond). Bonds associated to the aromatic ring and bonds of higher order contribute strongly to the predicted energy, whereas regions involving single bonds contribute negatively. To verify whether this observation generalizes to other molecules, we perform the GNN-LRP analysis on a set of 1,000 molecules randomly drawn from the QM9 dataset and show in Fig. 10 (right) the average bond contribution for each bond type. We observe an increasing energy contribution with ascending bond order. This coincides with chemical intuition that bonds of higher order are more stable and, thus, require more energy to break.—Note that the SchNet does not take bond types as an input, but as highlighted by GNN-LRP, it has clearly inferred these chemical features from the data.

We now turn to another quantum chemical property, the *dipole moment*, and its prediction by SchNet (cf. [76]). The quantity produced at the output of the model has the form $\|\boldsymbol{\mu}(\boldsymbol{\Lambda})\|$. The nonlinearity of the norm introduces higher-order terms in the GNN function and this prevents a direct application of GNN-LRP. Instead, we consider for explanation the dot product $\langle \boldsymbol{\mu}(\boldsymbol{\Lambda}), [\boldsymbol{\mu}(\boldsymbol{\Lambda})/\|\boldsymbol{\mu}(\boldsymbol{\Lambda})\|]_{\text{cst.}} \rangle$, where the left hand side functionally depends on the GNN input $\boldsymbol{\Lambda}$, and where the right hand side is the normalized direction of the predicted dipole moment, detached from the gradient computation. With this modification, while the output of the model remains the same locally, the top layer becomes linear, therefore GNN-LRP can proceed as for the energy prediction case.

Fig. 11 (top) shows the GNN-LRP explanation of the predicted dipole moment for the same molecule as in the previous experiment. Here, we observe that the contributions to the dipole moment found by GNN-LRP form a gradient from

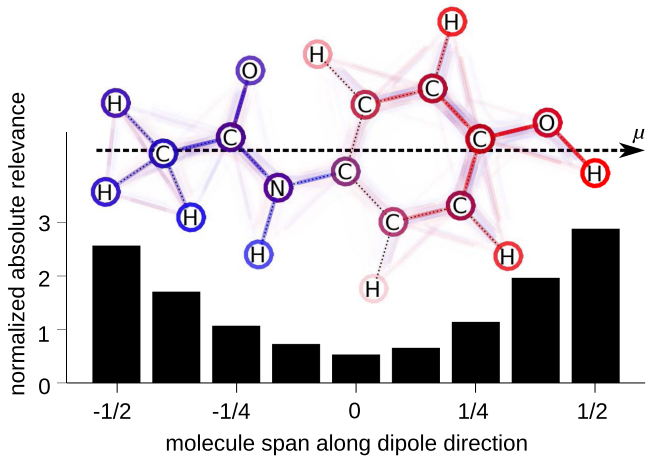


Fig. 11. *Top:* Paracetamol molecule with the predicted dipole direction shown as a dotted arrow, and the GNN-LRP explanation. *Bottom:* Distribution of contributions (in absolute terms) along the direction of the predicted dipole moment, averaged over the dataset.

one side to the other of the molecule. The orientation corresponds to the positive and negative pole of the molecule. To verify whether this insight generalizes to other molecules, we consider a set of 1,000 molecules and we normalize each molecule to a span of 1 along its dipole direction. Subsequently, we project all walks onto their respective dipole to obtain a one-dimensional distribution of absolute relevance values for all molecules. Note that the atom density of molecules, in general, is not homogeneous, and thus, in some regions more walks may occur while other regions do not exhibit as many walks. Hence, for each molecule the distribution of absolute relevance is normalized w.r.t. its atom density. The result of this aggregated analysis is shown in Fig. 11 (bottom).

This quantitative result confirms the alignment of GNN-LRP contributions with the positive and negative poles of the molecule, as it was found qualitatively on the paracetamol molecule. This result is in accordance with chemical intuition regarding the dependence of the dipole on the span of the molecule.

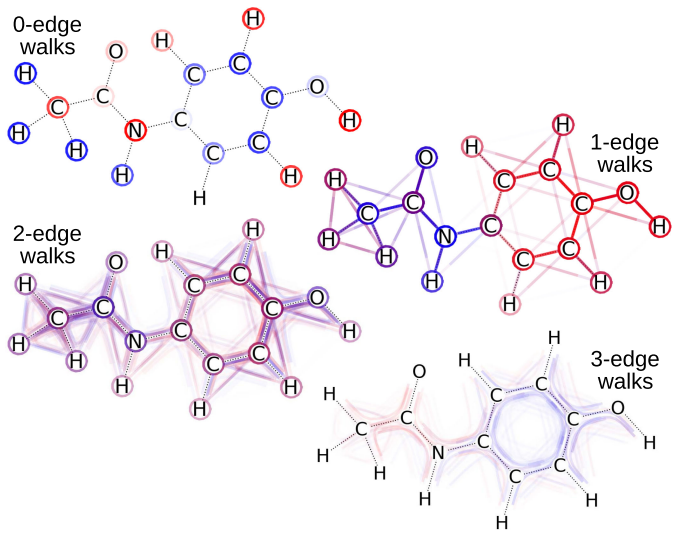


Fig. 12. Expanded GNN-LRP explanation for the dipole moment prediction of the paracetamol molecule. To increase visibility of two-edge walks and three-edge walks, we show the scaled relevance scores $R' = R^{0.7}$.

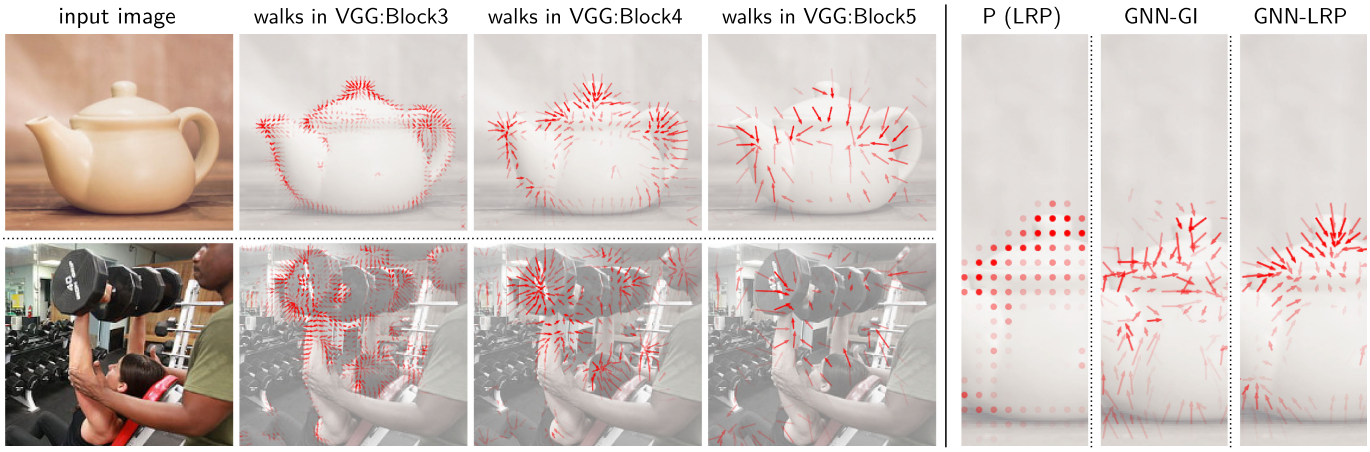


Fig. 13. *Left:* Relevant walks in the pixel lattice explaining the prediction by the VGG-16 network of two input images as 'teapot' and 'dumbbell' respectively. In each vector field, arrows connect block input nodes to the relevance-weighted average position of the block output nodes. *Right:* Comparison of GNN-LRP with different explanation techniques on Block 4.

Because the walk-based explanations produced by GNN-LRP are very detailed, some of the more intricate details of the explanation cannot always be visualized on a single molecule. Hence, we perform a further experiment where the GNN-LRP explanation is spread over multiple visuals, each of them showing relevant walks covering a specific number of edges. With this expanded visualization, we seek in particular to better distinguish between local atom-wise contribution and more global effects. Fig. 12 shows this expanded analysis of the dipole moment prediction for the paracetamol molecule.

From this visualization, we gain further insights into the strategy used by the SchNet model when predicting the dipole moment. In our expanded explanation, one-edge walks clearly indicate the electrostatic poles of the molecule, while giving a hint on local dipoles. Self-walks (i.e., 0-edge walks) incorporate elements that are inherent to the atom types, in particular, their electronegativity. Two-edge walks provide rather complex and spatially less resolved contributions. Finally, three-edge walks, that are also the most global descriptors, again provide interesting spatial contributions, and appear to dampen the one-edge walks contributions based on the more complex structures they are able to capture.

Overall, GNN-LRP has provided insights into the structure-property relationship of molecules that reach beyond the original prediction task. The resulting relevant walks agree with chemical characteristics of the molecule, thus, indicating that the neural network has indeed learned chemically plausible regularities.

### 5.3 Revisiting Image Classification

A convolutional neural network (CNN) can be seen as a particular graph neural network (GNN) operating on lattices of pixels. CNN predictions have so far mainly been explained using heatmaps highlighting pixels that are the most relevant for a given prediction [13], [16], [77]. Heatmaps are a useful representation summary of the decision structure, but they do not reveal the more complex strategies of a network that have been used to progressively build the prediction layer after layer. We will show by viewing CNNs as graph neural networks and extracting relevant walks in the resulting pixel lattice, that our GNN-LRP method is capable of shedding light into these strategies.

For this, we consider the well-established VGG-16 [49] network. It consists of a collection of blocks interleaved by pooling layers, where each block is composed of a sequence of convolution and ReLU layers. We use the pretrained version of the VGG-16 network [49] without batch-normalization, which can be retrieved using the TorchVision module of PyTorch.[6]

Because the VGG-16 neural network is deep and the number of possible walks grows exponentially with neural network depth, we marginalize explanations to only consider the position of the walk at the input and at the output of a block. This is easily achieved by using a mask-based implementation (cf. Appendix C of the Supplement, available online) and removing all masks except those at the input and output of the block. We then compute one explanation for block 3, 4, and 5. Also, to cope with the large spatial lattices in each block, we make use of the multi-mask strategy also outlined in Appendix C, available online. Specifically, observing that each block of the VGG-16 network has receptive fields of size 7, we can process multiple walks at the same time by choosing the mask to be a grid with stride 7. This allows us to collect all relevant walks at the given block in the order of 49 backward passes.

We consider two exemplary images[7] that the VGG-16 network respectively predicts as 'teapot' and 'dumbbell'. We set the LRP parameter to $\gamma = 0.5$ in block 3, halving the parameter value in each subsequent block, and choosing $\gamma = 0$ in the top-level classifier. Fig. 13 (left) shows the result of the analysis for these two images at various blocks of the VGG-16 network.

For the first image, Block 3 detects local edges in the teapot, then, in Block 4, the walks converge to center points of specific parts of the teapot (e.g., the handle, the spout and the knob), and finally the walks converge in Block 5 to the center of the teapot, which can be interpreted as composing

6. https://pytorch.org/vision/stable/models.html

7. Images are from https://www.piqsels.com/en/public-domain-photo-fjjsr and https://www.piqsels.com/en/public-domain-photo-fiffy, rescaled and cropped to the relevant region to produce images of size $224 \times 224$ which are the standard input size for VGG-16.

the different parts of the teapot. For this exemplary image, we further observe in Fig. 13 (right) the advantageous properties of GNN-LRP compared to more basic explanation methods. The GNN-GI baseline also produces a vector field, however, the latter is significantly more noisy than the one produced by GNN-LRP. The method by Pope *et al.* [24] robustly highlights relevant nodes at the input of the given block, however, it does not reveal where these features are being transported for use in the subsequent block.

For the second image, we investigate a known 'Clever Hans' strategy where the network classifies images as 'dumbbell' by detecting both the dumbbell and the arm that holds it [78]. Using GNN-LRP we observe that Blocks 3 and 4 detect the arm and the dumbbell separately, and then, Block 5 composes them into a single 'dumbbell-arm' concept, as shown by the walks for both objects converging to some center point near the wrist. Clearly these insights could not have been obtained from a standard pixel-wise heatmap explanation.

Overall, our GNN-LRP method can be used to comprehensively inspect the prediction of an image classifier beyond what would be possible with a standard pixel-wise heatmap explanation. This deeper explanation capability allows us to better understand the detailed structure of image classifications, and also to shed more light into anecdotal 'Clever Hans' effects observed in the context of state-of-the-art image classifiers.

## 6 Conclusion

Graph neural networks are a highly promising approach for predicting graphs, with a strong demand from the practical side. For these models to be broadly adopted, it is however important that their predictions are made explainable to the user.—Because the input of a GNN is tightly entangled with the model itself, the explanation problem is particularly difficult, and methods for explaining GNNs have so far been limited.

In this paper we have proposed a novel theoretically principled approach to produce these explanations, based on higher-order Taylor expansions. From this conceptual starting point, we have then contributed two practical algorithms: GNN-GI which we propose as a simple baseline, and GNN-LRP which is more robust and scales to highly non-linear models. GNN-LRP produces detailed explanations that subsume the complex nested interaction between the GNN model and the input graph. It also significantly outperforms other explanation methods in our quantitative benchmark.—In addition to the high quality of the explanations it produces, GNN-LRP is also broadly applicable (covering the GCN, the GIN, spectral filtering approaches and others), and it can handle virtually any type of input graph, whether it is a parse tree, a spatial graph, a pixel lattice, etc.

This broad applicability is demonstrated in our extensive application showcase, including sentiment analysis, quantum chemistry and image classification. In each scenario, GNN-LRP could highlight the diverse strategies employed by the GNN models, and unmask some undesired 'Clever Hans' strategies. In our quantum-chemical application showcase, we could additionally extract interesting problem-relevant insights. Future work will apply the proposed methodology to analyze properties of materials for practically highly relevant tasks, e.g., in catalysis.

## Acknowledgments

We would like to thank Jacob Kauffmann, Lukas Ruff and Marina Höhne for the highly helpful and illuminating discussions on the topic and also Niklas W. A. Gebauer for helping with the molecule visualization. We would like to thank Jasmijn Bastings for very helpful comments on the manuscript.

## References

[1] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[3] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet–A deep learning architecture for molecules and materials," *J. Chem. Phys.*, vol. 148, no. 24, 2018, Art. no. 241722.

[4] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[5] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1506–1515.

[6] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 1957–1967.

[7] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 273–283.

[8] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *Assoc. Comput. Machinery Trans. Graph.*, vol. 38, no. 5, pp. 146:1–146:12, 2019.

[9] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking Clever Hans predictors and assessing what machines really learn," *Nat. Commun.*, vol. 10, 2019, Art. no. 1096.

[10] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700. Berlin, Germany: Springer, 2019.

[11] F. Doshi-Velez and B. Kim, "A roadmap for a rigorous science of interpretability," *CoRR*, vol. abs/1702.08608, 2017.

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1135–1144.

[13] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 3319–3328.

[14] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1721–1730.

[15] D. Alvarez-Melis and T. S. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 7786–7795.

[16] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS One*, vol. 10, no. 7, 2015, Art. no. e0130140.

[17] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, "Explaining deep neural networks and beyond: A review of methods and applications," *Proc. IEEE*, vol. 109, no. 3, pp. 247–278, Mar. 2021.

[18] A. B. Arrieta *et al.*, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, 2020.

[19] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[20] O. Eberle, J. Buttner, F. Krautli, K.-R. Müller, M. Valleriani, and G. Montavon, "Building and interpreting deep similarity models," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Sep. 1, 2020, doi: 10.1109/TPAMI.2020.3020738.

[21] J. D. Janizek, P. Sturmfels, and S.-I. Lee, "Explaining explanations: Axiomatic feature interactions for deep networks," *J. Mach. Learn. Res.*, vol. 22, no. 104, pp. 1–54, 2021. [Online]. Available: http://jmlr.org/papers/v22/20–1223.html

[22] T. Cui, P. Marttinen, and S. Kaski, "Learning global pairwise interactions with Bayesian neural networks," in *Proc. 24th Eur. Conf. Artif. Intell.*, 2020, vol. 325, pp. 1087–1094.

[23] M. Tsang, D. Cheng, and Y. Liu, "Detecting statistical interactions from neural network weights," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[24] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10 772–10 781.

[25] R. Schwarzenberg, M. Hübner, D. Harbecke, C. Alt, and L. Hennig, "Layerwise relevance visualization in convolutional text graph classifiers," in *Proc. 13th Workshop Graph-Based Methods Natural Lang. Process.*, 2019, pp. 58–62.

[26] Z. Ying, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9240–9251.

[27] D. Luo *et al.*, "Parameterized explainer for graph neural network," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 19620–19631.

[28] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards model-level explanations of graph neural networks," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 430–438.

[29] M. Vu and M. Thai, "PGM-explainer: Probabilistic graphical model explanations for graph neural networks," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 12225–12235.

[30] M. S. Schlichtkrull, N. D. Cao, and I. Titov, "Interpreting graph neural networks for NLP with differentiable edge masking," in *Proc. Int. Conf. Learn. Representations*, 2021.

[31] W. Lin, H. Lan, and B. Li, "Generative causal explanations for graph neural networks," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 6666–6679.

[32] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, vol. 139, pp. 12 241–12 252.

[33] C. Ji, R. Wang, and H. Wu, "Perturb more, trap more: Understanding behaviors of graph neural networks," *CoRR*, vol. abs/2004.09808, 2020.

[34] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "GraphLIME: Local interpretable model explanations for graph neural networks," *CoRR*, vol. abs/2001.06216, 2020.

[35] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *CoRR*, vol. abs/2012.15445, 2020.

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[37] K. Schütt, P. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 991–1001.

[38] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3634–3640.

[39] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 1263–1272.

[40] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 22118–22133.

[41] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognit.*, vol. 65, pp. 211–222, 2017.

[42] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: An overview," in *Explainable AI*, vol. 11700. Berlin, Germany: Springer, 2019, pp. 193–209.

[43] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[44] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[45] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[46] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[47] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[48] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[50] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4805–4815.

[51] L. Arras *et al.*, "Explaining and interpreting LSTMs," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700. Berlin, Germany: Springer, 2019, pp. 211–238.

[52] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Modern Phys.*, vol. 74, no. 1, pp. 47–97, Jan. 2002.

[53] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2660–2673, Nov. 2017.

[54] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2013, pp. 1631–1642.

[55] N. Bansal, C. Agarwal, and A. M. Nguyen, "SAM: The sensitivity of attribution methods to hyperparameters," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 11–21.

[56] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *CoRR*, vol. abs/1605.01713, 2016.

[57] G. Montavon, "Gradient-based vs. propagation-based explanations: An axiomatic comparison," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700. Berlin, Germany: Springer, 2019, pp. 253–265.

[58] D. Balduzzi, M. Frean, L. Leary, J. P. Lewis, K. W. Ma, and B. McWilliams, "The shattered gradients problem: If resnets are the answer, then what is the question?," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, vol. 70, pp. 342–350.

[59] G. Montavon, W. Samek, and K.-R Müller, "Methods for interpreting and understanding deep neural networks," *Digit. Signal Process.*, vol. 73, pp. 1–15, 2018.

[60] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 9525–9536.

[61] B. Sanchez-Lengeling *et al.*"Evaluating attribution for graph neural networks," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 5898–5910.

[62] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, and P. Kessel, "Explanations can be manipulated and geometry is to blame," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, vol. 32, Art. no. 1217.

[63] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed.. Hoboken, NJ, USA: Prentice Hall, 2009.

[64] K. Rieck, T. Krueger, U. Brefeld, and K.-R. Müller, "Approximate tree kernels," *J. Mach. Learn. Res.*, vol. 11, no. 16, pp. 555–580, 2010.

[65] B. Liu, *Sentiment Analysis and Opinion Mining*. San Rafael, CA, USA: Morgan & Claypool, 2012.

[66] T. Bolukbasi, K. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4349–4357.

[67] H. Gonen and Y. Goldberg, "Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 609–614.

[68] O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, "Exploring chemical compound space with quantum-based machine learning," *Nat. Rev. Chem.*, vol. 4, pp. 347–358, 2020.
[69] F. Noé, A. Tkatchenko, K.-R. Müller, and C. Clementi, "Machine learning for molecular simulation," *Annu. Rev. Phys. Chem.*, vol. 71, no. 1, pp. 361–390, 2020.
[70] K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, *Machine Learning Meets Quantum Physics*, vol. 968. Berlin, Germany: Springer, 2020.
[71] J. A. Keith *et al.*, "Combining machine learning and computational chemistry for predictive insights into chemical systems," *Chem. Rev.*, vol. 121, no. 16, pp. 9816–9872, 2021.
[72] O. T. Unke *et al.*, "Machine learning force fields," *Chem. Rev.*, vol. 121, no. 16, pp. 10142–10186, 2021.
[73] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. -R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nat. Commun.*, vol. 8, 2017, Art. no. 13890.
[74] K. Schütt, P. Kessel, M. Gastegger, K. Nicoli, A. Tkatchenko, and K.-R. Müller, "SchNetPack: A deep learning toolbox for atomistic systems," *J. Chem. Theory Comput.*, vol. 15, no. 1, pp. 448–455, 2018.
[75] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Sci. Data*, vol. 1, 2014, Art. no. 140022.
[76] K. T. Schütt, M. Gastegger, A. Tkatchenko, and K.-R. Müller, "Quantum-chemical insights from interpretable atomistic neural networks," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700, W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds. Berlin, Germany: Springer, 2019, pp. 311–330.
[77] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, vol. 8689, pp. 818–833.
[78] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

**Thomas Schnake** received the MSc degree in mathematics from Humboldt Universität zu Berlin, Germany, in 2018, and the MSc degree in scientific computing from Technische Universität (TU) Berlin, Germany. He is currently working towards the PhD degree in the Machine Learning Group, TU Berlin, Germany with a research focus on explainable machine learning for structured data.

**Oliver Eberle** received the Joint MSc degrees in computational neuroscience from Technische Universität Berlin, Germany, and Humboldt Universität zu Berlin, Germany, in 2017. He is currently working towards the PhD degree in the Machine Learning Group, TU Berlin, Germany, and his research focuses on explainable machine learning and natural language processing.

**Jonas Lederer** received the MSc degree in condensed matter physics from Technische Universität München, Germany, in 2018. He is currently working towards the PhD degree in the Machine Learning Group, TU Berlin, Germany, and his research focuses on explainable machine learning and representation learning in quantum chemistry.

**Shinichi Nakajima** received the master's degree in physics from Kobe University, Japan, in 1995, and the doctoral degree in computer science from the Tokyo Institute of Technology, Japan, in 2006, and worked with Nikon Corporation, Japan, until September 2014 on statistical analysis, image processing, and machine learning. He is a senior researcher in BIFOLD, Technische Universität Berlin, Germany. His research interest is in theory and applications of machine learning, in particular, Bayesian learning theory, variational inference, generative models, computer vision, explainable AI, and machine learning applications for science.

**Kristof T. Schütt** received the master's degree in computer science, and the PhD degree in machine learning at the Machine Learning Group of Technische Universität Berlin, Germany, in 2012 and 2018, respectively. He is a senior researcher at the Berlin Institute for the Foundations of Learning and Data (BIFOLD), Germany. Until September 2020, he worked at the Audatic company developing neural networks for real-time speech enhancement. His research interests include interpretable neural networks, representation learning, generative models, and machine learning applications in quantum chemistry.

**Klaus-Robert Müller** (Member, IEEE) received the degree in physics in Karlsruhe, Germany in 1989, and the PhD degree in computer science at Technische Universität Karlsruhe, Germany, in 1992. He has been a professor of computer science at Technische Universität Berlin, Germany since 2006; at the same time he is co-directing the Berlin Big Data Center. After completing a postdoctoral position at GMD FIRST in Berlin, Germany, he was a research fellow at the University of Tokyo, Japan from 1994 to 1995. In 1995, he founded the Intelligent Data Analysis Group, GMD-FIRST (later Fraunhofer FIRST) and directed it until 2008. From 1999 to 2006, he was a professor at the University of Potsdam, Germany. He was awarded the Olympus Prize for Pattern Recognition (1999), the SEL Alcatel Communication Award (2006), the Science Prize of Berlin by the Governing Mayor of Berlin (2014), and the Vodafone Innovations Award (2017). In 2012, he was elected member of the German National Academy of Sciences-Leopoldina, in 2017 of the Berlin Brandenburg Academy of Sciences and also in 2017 external scientific member of the Max Planck Society. In 2019 and 2020 he became ISI Highly Cited researcher. His research interests are intelligent data analysis and machine learning with applications in neuroscience (specifically brain-computer interfaces), physics and chemistry.

**Grégoire Montavon** received the master's degree in communication systems from École Polytechnique Fédérale de Lausanne, Switzerland, in 2009, and the PhD degree in machine learning from the Technische Universität Berlin, Germany, in 2013. He is a senior researcher in the Machine Learning Group, Technische Universität Berlin, Germany, and in the Berlin Institute for the Foundations of Learning and Data (BIFOLD). He is a member of the ELLIS Unit Berlin, and an editorial board member of Pattern Recognition. He is recipient of the 2020 Pattern Recognition Best Paper Award. His research interests include explainable machine learning, deep neural networks, and unsupervised learning.