Explainability Methods for Graph Convolutional Neural Networks

Phillip E. Pope*
HRL Laboratories, LLC
pepope@hrl.com

Soheil Kolouri*
HRL Laboratories, LLC
skolouri@hrl.com

Mohammad Rostami
HRL Laboratories, LLC
mrostami@hrl.com

Charles E. Martin
HRL Laboratories, LLC
cemartin@hrl.com

Heiko Hoffmann
HRL Laboratories, LLC
hhoffmann@hrl.com

Abstract

With the growing use of graph convolutional neural networks (GCNNs) comes the need for explainability. In this paper, we introduce explainability methods for GCNNs. We develop the graph analogues of three prominent explainability methods for convolutional neural networks: contrastive gradient-based (CG) saliency maps, Class Activation Mapping (CAM), and Excitation Backpropagation (EB) and their variants, gradient-weighted CAM (Grad-CAM) and contrastive EB (c-EB). We show a proof-of-concept of these methods on classification problems in two application domains: visual scene graphs and molecular graphs. To compare the methods, we identify three desirable properties of explanations: (1) their importance to classification, as measured by the impact of occlusions, (2) their contrastivity with respect to different classes, and (3) their sparseness on a graph. We call the corresponding quantitative metrics fidelity, contrastivity, and sparsity and evaluate them for each method. Lastly, we analyze the salient subgraphs obtained from explanations and report frequently occurring patterns.

1. Introduction

Recent success in computer vision is mainly due to emergence of deep convolutional neural networks (CNNs) [21]. This has led to state-of-the-art performances on various computer vision tasks including object recognition [11, 13], object detection [27], and semantic segmentation [26]. The end-to-end nature of learning in CNNs have turned them into powerful data-driven tools for learning from a large corpus of visual data. At the same time, this end-to-end learning strategy hinders the explainability and interpretability of decisions made by CNNs. Recently, there has been an increasing number of works studying the inner workings of CNNs [38, 23, 22] and explaining the decisions made by these networks [42, 31, 39, 40]. Zhang et al. [41] provide a good survey on methods for explainability of CNNs.

Deep CNNs, however, are designed for grid structured data, e.g. images, in Euclidean spaces, as convolution is an operation defined on Euclidean space for inputs with ordered elements. Nonetheless, in many applications we need to deal with data defined on different structures, e.g. graphs and manifolds, where CNNs cannot be directly used. Such non-Euclidean spaces appear in various applications including scene graph analysis [15], 3D-shape analysis [25], social networks [37], and chemistry [35]. Geometric deep learning [6, 1] is a recent emerging field to overcome limitations of CNNs and broaden their application. In particular, CNNs could be generalized to be applicable on graph-structured data by extending the convolution operation onto graphs and in general onto non-Euclidean spaces. The extension of CNNs to non-Euclidean spaces leads to graph convolution neural networks (GCNNs) [7, 9, 19].

In addition to superior performance of a model, we need techniques to explain why a model predicts what it predicts. This explanation can help to identify and localize parts of the input data relevant to the model's decisions in a particular task. Inspired by the explainability work on CNNs [42], we introduce explainability methods for decisions by GCNNs. Explainability can be particularly helpful for graphs, even more than for images, because non-expert humans cannot intuitively determine the relevant context within a graph, for example, when identifying groups of atoms (a sub-graph structure on a molecular graph) that contribute to a particular property of a molecule.

We adapt three common explainability methods, originally designed for CNNs, and extend them to GCNNs. These three methods are gradient-based saliency maps [32], Class Activation Mapping (CAM) [39], and Excitation Backpropagation (EB) [39]. In addition, we adapt two variants: gradient-weighted CAM (Grad-CAM) [31] and contrastive EB. We evaluate the adapted methods on two differ-

ent applications: visual scene graphs and molecular graphs. For GCNNs, we use the proposed formulation by Kipf et al. [18]. Our specific contributions in this work are the following three:

- Adaptation of explainability methods for CNNs to GCNNs,
- Demonstration of the explainability techniques on two graph classification problems: visual scene graphs and molecules, and
- Characterization of each method's trade-offs using metrics for fidelity, contrastivity, and sparsity.

The remainder of this paper is structured as follows. In Section 2, we discuss related work in interpretability and GCNNs. In Section 3, we review the mathematical definitions of GCNNs and explainability methods on CNNs, and then define the analogous explainability methods on GCNNs. In Section 4, we detail our experiments on visual scene graphs and molecules and show example results. Moreover, we quantitatively evaluate the performance of these four methods with respect to three metrics, fidelity, contrastivity, and sparsity, each designed to capture certain desirable properties of explanations. We use these metrics to evaluate the merits of each method. Lastly, in the experimental section, we analyze the frequencies of salient substructures identified by Grad-CAM and report the top results for each dataset.

2. Related Work

Interpretability: A long standing limitation of general deep neural networks has been the difficulty in interpreting and explaining the classification results. Recently, explainability methods have been devised for deep networks and specifically CNNs [32, 42, 31, 39, 40, 41]. These methods enable one to probe a CNN and identify the important substructures of the input data (as deemed by the network) for decision regarding a task, which could be used as an explanatory tool or as a tool to discover unknown underlying substructures in the data. For example, in the area of medical imaging [34], in addition to classifying images having malignant lesions, they can be localized, as the CNN can provide reasoning for classifying an input image.

The most straightforward approach for generating a sensitivity map over the input data to discover the importance of the underlying substructures is to calculate a gradient map within a layer by considering the norm of the gradient vector with respect to an input for each network weight [32]. However, gradient maps are known to be noisy and smoothing these maps might be necessary [33]. More advanced techniques include Class Activation Mapping (CAM) [42], Gradient-weighted Class Activation Mapping (Grad-CAM) [31], and Excitation Back-Propagation (EB) [39] improve gradient maps by taking into account some notion of context. These techniques have been shown to be effective on CNNs and can identify highly abstract notions in images. See Zhang et. al. [41] for a survey of explainability methods for CNNs.

Graph Convolutional Neural Networks: The mathematical foundation of GCNNs is deeply rooted in the field of graph signal processing [3, 4] and spectral graph theory in which signal operations like Fourier transform and convolutions are extended to signals living on graphs. GCNNs emerged from the spectral graph theory, e.g., as introduced by Bruna et al. [2] or Henaff et al. [12]. GCNNs based on spectral graph theory enable definition of parameterized filters akin to CNNs. They, however, are often computationally expensive and therefore slow. To overcome the computational bottleneck of spectral GCNNs, various authors have proposed to approximate smooth filters in the spectral domain [6, 19], for instance using Chebyshev polynomials or a first-order approximation of spectral graph convolutions. In this work, we use the GCNN formulation defined by Kipf and Welling [19] due to its faster training times and higher predictive accuracy.

GCNNs have recently found use in diverse applications. Monti et al. [25] used GCNNs for super-pixel classification as well as for classifying research papers from their citation network. Defferard et al. [6] used GCNNs on N-grams for text categorization. In [36] GCNNs were used for shape segmentation, and in [14], they were used for skeleton-based action recognition. More recently, Johnson et al. [15] used GCNNs to analyze scene-graphs with the application of image generation from scene graphs. In chemistry, GCNNs were used to predict various chemical properties of organic molecules. GCNNs provide state-of-the-art performance on several chemical prediction tasks, including toxicity prediction [16], solubility [7], and energy prediction [30]. In this paper we focus on explainability methods for GCNNs with applications on scene graph classification and molecule classification.

3. Methods

We compare and contrast the application of popular explainability methods to Graph Convolutional Neural Networks (GCNNs). Furthermore, we explore the benefits of a number of enhancements to these approaches.

3.1. Explainability for CNNs

The three main groups of popular explainability methods are contrastive gradients, Class Activation Mapping, and Excitation Backpropagation.

Contrastive gradient-based saliency maps [32] is perhaps the most straight-forward and well-established approach. In this approach, one simply differentiates the output of the model with respect to the model input, thus cre-

ating a heat-map, where the norm of the gradient over input variables indicates their relative importance. The resulting gradient in the input space points in the direction corresponding to the maximum positive rate of change in the model output. Therefore, the negative values in the gradient are discarded to only retain the parts of input that positively contribute to the solution:

$$L^c_{Gradient} = \| \text{ReLU}\left(\frac{\partial y^c}{\partial x}\right) \| \quad , \tag{1}$$

where y^c is the score for class c before the softmax layer, and x is the input. While easy to compute and interpret, saliency maps generally perform worse than newer techniques (like CAM, Grad-CAM, and EB), and it was recently argued that saliency maps tend to represent noise rather than signal [17].

Class Activation Mapping provides an improvement over saliency maps for convolutional neural networks, including GCNNs, by identifying important, class-specific features at the last convolutional layer as opposed to the input space. It is well-known that such features tend to be more semantically meaningful (e.g., faces instead of edges). The downside of CAM is that it requires the layer immediately before the softmax classifier (output layer) to be a convolutional layer followed by a global average pooling (GAP) layer. This precludes the use of more complex, heterogeneous networks, such as those that incorporate several fully connected layers before the softmax layer.

To compute CAM, let $F_k \in \mathbb{R}^{u \times v}$ be the k^{th} feature map of the convolutional layer preceding the softmax layer. Denote the global average pool (GAP) of F_k by

$$e_k = \frac{1}{Z} \sum_i \sum_j F_{k,i,j} \tag{2}$$

where $Z = uv$. Then, a given class score, y^c, can be defined as

$$y^c = \sum_k w^c_k e_k, \tag{3}$$

where the weights w^c_k are learned based on the input-output behavior of the network. The weight w^c_k encodes the importance of feature k for predicting class c. By upscaling each feature map to the size of the input images (to undo the effect of pooling layers) the class-specific heat-map in the pixel-space becomes

$$L^c_{CAM}[i,j] = \text{ReLU}\left(\sum_k w^c_k F_{k,i,j}\right). \tag{4}$$

The Grad-CAM method improves upon CAM by relaxing the architectural restriction that the penultimate layer must be a convolutional. This is achieved by using feature map weights α^c_k that are based on back-propagated gradients. Specifically, Grad-CAM defines the weights according to

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial F_{k,i,j}}. \tag{5}$$

Following the intuition behind Equation (4) for CAM, the heat-map in the pixel-space according to Grad-CAM is computed as

$$L^c_{Grad-CAM}[i,j] = \text{ReLU}\left(\sum_k \alpha^c_k F_{k,i,j}\right), \tag{6}$$

where the ReLU function ensures that only features that have a *positive* influence on the class prediction are non-zero.

Excitation Backpropagation is an intuitively simple, but empirically effective explanation method. In [28], it is argued and demonstrated experimentally that explainability approaches such as EB [39], which ignore nonlinearities in the backward-pass through the network, are able to generate heat-maps that "conserve" evidence for or against a network predicting any particular class. Let a^l_i be the i'th neuron in layer l of a neural network and $a^{(l-1)}_j$ be a neuron in layer $(l-1)$. Define the *relative* influence of neuron $a^{(l-1)}_j$ on the activation $y^l_i \in \mathbb{R}$ of neuron a^l_i, where $y^l_i = \sigma(\sum_{ji} W^{l-1}_{ji} y^{(l-1)}_j)$ and for $W^{(l-1)}$ being the synaptic weights between layers $(l-1)$ and l, as a probability distribution $P(a^{(l-1)}_j)$ over neurons in layer $(l-1)$. This probability distribution can be factored as

$$P(a^{(l-1)}_j) = \sum_i P(a^{(l-1)}_j | a^l_i) P(a^l_i). \tag{7}$$

Zhang et al. [39] then define the conditional probability $P(a^{(l-1)}_j | a^l_i)$ as

$$P(a^{(l-1)}_j | a^l_i) = \begin{cases} Z^{(l-1)}_i y^{(l-1)}_j W^{(l-1)}_{ji} & \text{if } W^{(l-1)}_{ji} \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where

$$Z^{(l-1)}_i = \left(\sum_j y^{(l-1)}_j W^{(l-1)}_{ji}\right)^{-1}$$

is a normalization factor such that $\sum_j P(a^{(l-1)}_j | a^l_i) = 1$. For a given input (e.g., an image), EB generates a heat-map in the pixel-space w.r.t. class c by starting with $P(a^L_i = c) = 1$ at the output layer and applying Equation (7) recursively.

These reviewed explainability methods were originally designed for CNNs, which are defined for signals on a uniform square grid. Here, we are interested in signals supported on non-Euclidean structures, e.g., graphs. In what

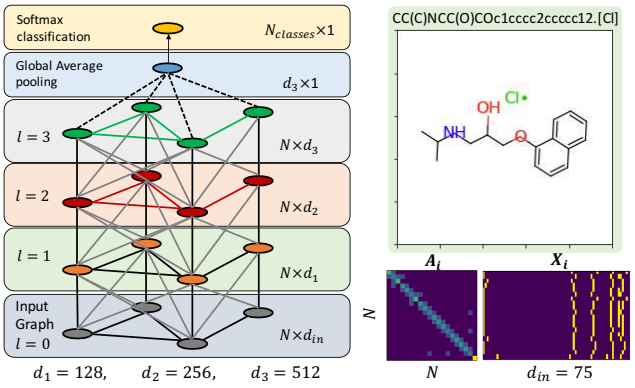

Figure 1: Our GCNN + GAP architecture together with the visualization of the input feature and adjacency matrix for a sample molecule from the BBBP dataset.

follows, we first briefly discuss GCNNs and then describe the extensions of these explainability methods to GCNNs. For intuition, we note that images may be conceptualized as lattice-shaped graphs with pixel values as node feature. In this sense, GCNNs generalize CNNs to accommodate arbitrary connectivity between nodes.

3.2. Graph Convolutional Neural Networks

Let an attributed graph with N nodes be defined with its node attributes $X \in \mathbb{R}^{N \times d_{in}}$ and its adjacency matrix $A \in \mathbb{R}^{N \times N}$ (weighted or binary). In addition, let the degree matrix for this graph be $D_{ii} = \sum_j A_{ij}$. Following the work of Kipf and Welling [19], we define the graph convolutional layer to be

$$F^l(X, A) = \sigma(\underbrace{\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}}_{V} F^{(l-1)}(X, A)W^l)\,, \qquad (9)$$

where F^l is the convolutional activations at the $l'th$ layer, $F^0 = X$, $\tilde{A} = A + I_N$ is the adjacency matrix with added self connections where $I_N \in \mathbb{R}^{N \times N}$ is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^l \in \mathbb{R}^{d_l \times d_{l+1}}$ are the trainable convolutional weights, and $\sigma(\cdot)$ is the element-wise nonlinear activation function. Figure 1 shows the used GCNN architecture in this work, where the activations in layers $l = 1, 2, 3$ follow Eq. (9), which is a first-order approximation of localized spectral filters on graphs.

For molecule classification, each molecule can be represented as an attributed graph $\mathcal{G}_i = (X_i, A_i)$, where the node features X_i summarize the local chemical environment of the atoms in the molecule, including atom-types, hybridization types, and valence structures [35], and the adjacency matrix encodes atomic bonds and demonstrate the connectivity of the whole molecule (see Figure 1). For a given dataset of labeled molecules $\mathcal{D} = \{\mathcal{G}_i = (X_i, A_i), y_i\}_{i=1}^M$ with labels y_i indicating a certain chemical property, e.g., blood-brain-barrier penetrability or toxicity, the task is to learn a classifier that maps each molecule to its corresponding label, $g : (X_i, A_i) \rightarrow y_i$.

Given that our task is to classify individual graphs (i.e., molecules) with potentially different number of nodes, we use several layers of graph convolutional layers followed by a global average pooling (GAP) layer over the graph nodes (e.g., atoms). In this case, all graphs will be represented with a fixed size vector. Finally, the GAP features are fed to a classifier. To enable applicability of CAM [42], we simply used a softmax classifier after the GAP layer.

3.3. Explainability for Graph Convolutional Neural Networks

In this subsection, we describe the extension of CNN explainability methods to GCNNs. Let the k'th graph convolutional feature map at layer l be defined as:

$$F_k^l(X, A) = \sigma(V F^{(l-1)}(X, A)W_k^l) \qquad (10)$$

where W_k^l denotes the $k'th$ column of matrix W^l, and $V = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ (see Eq. (8)). In this notation, for node n, the k'th feature at the l'th layer is denoted by $F_{k,n}^l$. Then, the GAP feature after the final convolutional layer, L, is calculated as

$$e_k = \frac{1}{N}\sum_{n=1}^{N} F_{k,n}^L(X, A) \quad , \qquad (11)$$

and the class score is calculated as, $y^c = \sum_k w_k^c e_k$. Using this notation, we extend the explainability methods to GCNNs as follows:

Gradient-based heat-maps over nodes n are

$$L_{Gradient}^c[n] = \|\text{ReLU}\left(\frac{\partial y^c}{\partial X_n}\right)\| \quad . \qquad (12)$$

CAM heat-maps are calculated as

$$L_{CAM}^c[n] = \text{ReLU}(\sum_k w_k^c F_{k,n}^L(X, A)))\,. \qquad (13)$$

Grad-CAM's class specific weights for class c at layer l and for feature k are calculated by

$$\alpha_k^{l,c} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial y^c}{\partial F_{k,n}^l} \quad , \qquad (14)$$

and the heat-map calculated from layer l is

$$L_{Grad-CAM}^c[l, n] = \text{ReLU}(\sum_k \alpha_k^{l,c} F_{k,n}^l(X, A))\,. \qquad (15)$$

Grad-CAM enables us to generate heat-maps with respect to different layers of the network. In addition, for our model shown in Figure 1, Grad-CAM's heat-map at the final convolutional layer and CAM's heat-map are equivalent

$L^c_{Grad-CAM}[L,n] = L^c_{CAM}[n]$ (See [31] for more details). In this work, we report results for $L^c_{Grad-CAM}[L,n]$ and the variant Grad-CAM Avg defined by

$$L^c_{Grad-CAMAvg}[n] = \frac{1}{L}\sum_{l=1}^{L} L^c_{Grad-CAM}[l,n]\,. \quad (16)$$

Excitation Backpropagation's heat-map for our model is calculated via backward passes through the softmax classifier, the GAP layer, and several graph convolutional layers. The equations for backward passes through the softmax classifier and the GAP layer are

$$\begin{cases} p(e_k) = \sum_c \frac{e_k ReLU(w^c_k)}{\sum_k e_k ReLU(w^c_k)} p(c) & \text{Softmax} \\ p(F^L_{k,n}) = \frac{F^L_{k,n}}{N e_k} p(e_k) & \text{GAP}\,, \end{cases} \quad (17)$$

where $p(c) = 1$ for the class of interest and zero otherwise. The backward passes through the graph convolutional layers, however, are more complicated. For notational simplicity, we decompose a graph convolutional operator into

$$\begin{cases} \hat{F}^l_{k,n} = \sum_m V_{n,m} F^l_{k,m} \\ F^{(l+1)}_{k',n} = \sigma(\sum_{k'} \hat{F}^l_{k,n} W^l_{k,k'})\,, \end{cases} \quad (18)$$

where the first equation is a local averaging of atoms (with $V_{n,m} \geq 0$), and the second equation is a fixed perceptron applied to each atom (analogous to one-by-one convolutions in CNNs). The corresponding backward passes for these two functions can be defined as

$$\begin{cases} p(F^l_{k,n}) = \sum_m \frac{V_{n,m} F^l_{k,n}}{\sum_n V_{n,m} F^l_{k,m}} p(\hat{F}^l_{k,m}) \\ p(\hat{F}^l_{k,n}) = \sum_{k'} \frac{\hat{F}^l_{k,n} ReLU(W^l_{k,k'})}{\sum_k \hat{F}^l_{k,n} ReLU(W^l_{k,k'})} p(F^{(l+1)}_{k',n})\,. \end{cases} \quad (19)$$

We generate the heat-map over the input layer by recursively backpropagating through the network and averaging the backpropagated probability heat-maps on the input:

$$L^c_{EB}[n] = \frac{1}{d_{in}} \sum_{k=1}^{d_{in}} p(F^0_{k,n}) \quad . \quad (20)$$

The contrastive extension of L^c_{EB} follows Eq. (8) in [39]; we call this contrastive variant, c-EB.

4. Experiments

This section describes our experiments and analysis of class-specific explanations. We experiment on two application domains, visual scene graphs and molecules. Additionally, we perform a frequency analysis of graph substructures identified by the explainability methods in the supplementary material.

4.1. Explanations on Scene Graphs

Our goal here is to train GCNNs for scene-graph classification and use our proposed explainability methods on these GCNNs. A scene graph is a graph structured data where nodes are objects in the scene and edges are relationships between objects. We obtain the data for our first experiment from the Visual Genome dataset [20]. The Visual Genome dataset consists of images and scene graph pairs. Objects and relationships are of many types and the data is collected from free text responses obtained from crowd sourced workers. Objects have an associated region of the image, defined by a bounding box.

We construct two binary classifications tasks on scene graphs from the Visual Genome: country vs. urban, and indoor vs outdoor. We model each of these words with a set of keywords, which are used to query the Visual Genome data for matches in any attribute of an image. The image set for a class is union of the image sets returned for each keyword. In addition, any intersection between classes pairs was removed. Keywords used to define each class are as follows:

- "country": country, countryside, farm, rural, cow, crops, sheep
- "urban": urban, city, downtown
- "indoor": indoor, room, office, bedroom, bathroom
- "outdoor": outdoor, nature, outside

The keywords are non-comprehensive and are not claimed to encompass the full meaning of each word. Rather they are synthetic constructions devised for the purpose of studying explanation methods on graphs. Since the aim of this work is to study explanations, the exact details of the class definitions are not germane to the present work. Keywords were chosen to give approximately balanced classes between classification pairs. Class ratios for each dataset are reported in Table 1.

For simplicity, and without loss of generality, we collapse all relationships into single type, which represents a generic relationship between two objects. We note that the various extensions of GCNNs exist for graphs with relational edges [29] for which the explainability methods developed in this paper could be used.

There is a bounding box associated with each object (i.e., node) in the scene-graph. We use a pretrained InceptionV3 network and extract deep features of the underlying image for bounding boxes for all the nodes of a scene-graph. The final pooling layer was used as the visual feature, where each crop was zero padded to a fixed size. The feature extracted from each bounding box is of size $d = 2048$. Thus our derived scene graphs consist of relational and visual data. The final task is then to classify the scene-graphs into classes urban vs. country and indoor vs. outdoor.

4.2. Explanations on Molecular Graphs

We study a second application domain: identifying functional groups on organic molecules for biological molecular properties. We evaluated explanation methods on three binary classification molecular datasets, BBBP, BACE, and task NR-ER from TOX21 [35]. Each dataset contains binary classifications of small organic molecules as determined by experiment. The BBBP dataset contains measurements on whether a molecule permeates the human blood brain barrier and is of significant interest to drug design. The BACE dataset contains measurements on whether a molecule inhibits the human enzyme β-secretase. The TOX21 dataset contains measurements of molecules for several toxicity targets. We selected the NR-ER task from this data, which is concerned with activation of the estrogen receptor [24]. These datasets are imbalanced. See Table 1 for the class ratios,

In addition, we followed the recommendations in [35], which is the original paper describing the MoleculeNet dataset, for train/test partitioning. In particular, for BACE and BBBP, the so called "scaffold" split is recommended by [35], which partitions molecules according to their structure, i.e. structurally similar molecules are partitioned in the same split. We emphasize that training the GCNNs and the conventional dataset splits are not the contribution of our paper and we simply follow the standard practice for these datasets.

4.3. Training and Evaluation

We partition the datasets into 80:20 training/testing sets. For all datasets, we used the GCNN + GAP architecture as described in Figure 1 with the following configuration: three graph convolutional layers of size 128, 256, and 512, respectively, followed by a GAP layer, and a softmax classifier. Models were trained for 25 epochs using the ADAM optimizer with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The models were implemented in Keras with Tensorflow backend [5]. Evaluation metrics AUC-ROC and AUC-PR of trained models for each dataset are reported in Table 1.

Some molecular classifications results are observed to have higher mean testing vs. training performance. Although this is unusual, the results are consistent to those reported in [35].

4.4. Analysis of Explanation Methods

After training models for each dataset, we apply each explanation method on all samples and to obtain a set of scalars over nodes, i.e. a heatmap.

We show selected results in Figures 3 and 4. In Figure , scalar importance values are encoded as the color of the bounding box outline (red: low, yellow: middle, green: high). In Figure 4, scalar importance values are encoded as the intensity of blue disk over each atom (white: low, blue: high).

Dataset	N_p	N_n	AUC-ROC		AUC-PR	
			Train	Test	Train	Test
Country vs. Urban	3267	3862	0.987	0.986	0.939	0.936
Indoor vs. Outdoor	8290	7971	0.962	0.963	0.871	0.876
BBBP	1560	470	0.991	0.966	0.994	0.966
BACE	691	821	0.994	0.966	0.939	0.999
TOX21	793	5399	0.868	0.881	0.339	0.347

Table 1: Class breakdown and evaluation metrics for each dataset. N_p and N_n denote the number of positive and negative examples respectively.

The heat-maps are calculated for positive and negative classes and normalized for each molecule across both classes and nodes to form a probability distribution over the input nodes. Class specificity can be seen by comparing explanations across classes within a method, i.e., when nodes activated by one class tend to be inactivate for the other.

Next, we report three quantitative metrics that capture desirable aspects of explanations: fidelity, contrastivity, and sparsity.

Fidelity was calculated to capture the intuition that occlusion of salient features identified through explanations should decrease classification accuracy. More precisely, we define fidelity as the difference in accuracy obtained by occluding all nodes with saliency value greater than 0.01 (on a scale 0 to 1). We then averaged the fidelity scores across classes for each method. We report these values in Table 2. The Contrastive Gradient method showed highest fidelity.

Contrastivity was designed to capture the intuition that class-specific features highlighted by an explanation method should differ between classes. More precisely, we define contrastivity as the ratio of the Hamming distance d_H between binarized heat-maps $\hat{m_0}, \hat{m_1}$ for positive and negative classes, normalized by the total number of atoms identified by either method, $\hat{m_0} \vee \hat{m_1}$, $\frac{d_H(\hat{m_0},\hat{m_1})}{\hat{m_0} \vee \hat{m_1}}$. We report this metric in Table 2. Grad-CAM showed the highest contrastivity.

Sparsity was designed to measure the localization of an explanation. Sparse explanations are particularly useful for studying large graphs, where manual inspection of all nodes is infeasible. More precisely, we define this measure as one minus the number of identified objects in *either* explanation $\hat{m_0} \vee \hat{m_1}$, divided by the total number of nodes in the graph $|V|$, $1 - \frac{\hat{m_0} \vee \hat{m_1}}{|V|}$. We report these values in Table 2. The c-EB method showed the sparsest activations.

See the Supplementary for a descriptive Figure of contrastivity and sparsity.

Using these metrics, we compare the quality of each explanation method and demonstrate the trade-offs of each method. In short, we conclude three main points: (1) Grad-

CAM is the most contrastive, has the second highest fidelity, but low sparsity. This method is generally suitable, but may be problematic on large graphs. (2) c-EB is the most sparse but has low fidelity and the second highest contrastivity. Therefore, this method is most suitable for analyzing large graphs at the expense of low fidelity. (3) Contrastive gradient (CG) has the highest fidelity, but low sparsity and low contrastivity. The lack of contrastivity makes this method unsuitable for class-specific explanations.

4.5. Subgraph Frequency Analysis

As discussed above, explanation methods on GCNNs identify salient structure on input graphs. The collection of salient substructures identified in a dataset suggests an opportunity for further analysis. We analyze this collection for reoccurring substructures, with the goal of yielding further insight into the data. In short, we compute the prevalence of each salient structure in the dataset, and compare against its prevalence in the positively labeled dataset. We give a brief description below, and refer the reader to the Supplementary for further details.

To identify salient substructure in a graph, we first collect the set of vertices with saliency value greater than some threshold (here, 0), which we call *activated vertices*. Then we collect the connected components induced by this vertex set. Repeating this for all graphs in the dataset yields a collection of salient subgraphs. We then count the frequency of each identified subgraph s in (1) the explanation set, (2) the positively labeled data, and (3) the negatively data, which we denote as N_e^s, N_p^s and N_n^s respectively. We used these counts to normalize the counts obtained from the explanations and construct three ratios: $R_e^s = \frac{N_e^s}{N_p^s+N_n^s}$, $R_p^s = \frac{N_p^s}{N_p^s+N_n^s}$, $R_n^s = \frac{N_n^s}{N_p^s+N_n^s}$. The ratio R_e^s measures the prevalence of a subgraph in explanations. The ratios R_p^s, R_n^s measure how prevalent a substructure occurs in positively and negatively labeled data respectively, and serve as a baseline for the first. Note that high R_p^s or R_n^s corresponds to high class specificity for salient subgraph s.

We report the top structures by the ratio R_e^s obtained from Grad-CAM for the molecule datasets and Visual Genome datasets in Figure 2.

In the case of molecules, we note that salient subgraphs have an interpretation as *functional groups*, i.e. a substructures that have a functional role in determining a molecular property, e.g. toxicity. Top results in Figure 2 include amides, tricholoromethyl, sulfonamides, and aromatic structures. The validity of these results as functional groups remains to be determined by experiment.

Other work in algorithmic identification of functional groups include [10] and [8]. Both are rule based approaches, whereas our method is learned. [8] discusses some of the limitations of algorithmic identification of functional groups. The notion of functional group is not precisely defined. [8] remarks that every medicinal chemists likely has their own notion of functional group. We do not claim our method is the right definition of functional groups. However, given the success of convolution neural networks in modeling other intuitive concepts in vision and language, we envision potential for practical applications in this area. Further subgraph results are shown in the Supplementary.

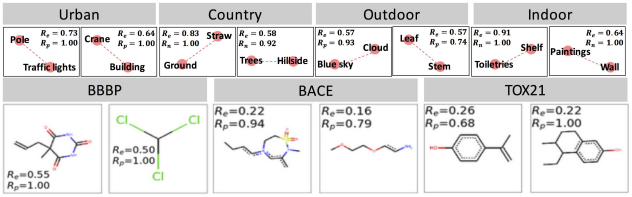

Figure 2: Top two substructures for each class/property, as identified by CAM/GradCAM.

5. Conclusions

In this work, we extended explainability methods designed for CNNs to GCNNs and compared these methods qualitatively and quantitatively. We demonstrated the extended methods on two application domains: visual scene graphs and molecular graphs.

We characterized the explanations obtained by each method with three metrics: fidelity, contrastivity, and sparsity. Grad-CAM showed the highest contrastivity, followed by c-EB. While c-EB showed the sparsest activations. Finally, CG showed the highest fidelity, but lowest contrastivity and second lowest sparsity. When finding class-specific explanations, however, contrastivity is a key requirement, which would disqualify CG.

We conclude that Grad-CAM is most suitable among the studied methods for explanations on graphs of moderate size. However, we caution the reader to choose a method with characteristics most appropriate to their application. If sparsity of explanations of is high concern, e.g., in the case of interpreting large graphs, then c-EB might be the better choice.

On both scene graphs and molecules, we identified graph substructures that were relevant for a given classification, e.g., identifying a scene as being indoor. Such a method may be useful for automatically mining relevant functional groups of molecules. Additional experiments, however, are needed to confirm the chemical validity of the identified substructures.

This work demonstrates tools for explaining GCNNs and mining patterns in substructures of graph inputs, including scene graphs, molecules, social networks, and electrical grids. In future work, we plan to investigate new explainability methods for graphs, and the use of explainability to improve the quality of visual scene graphs.

	Country vs. Urban	Indoor vs. Outdoor	BBBP	BACE	TOX21	
CG	0.40	0.40	0.19	0.38	0.53	**Fidelity**
	0.02 ± 0.45	0.18 ± 1.88	0.45± 2.19	0.77± 2.99	0.2 ± 2.13	**Contrastivity**
	11.67 ± 16.42	14.23 ± 17.45	0.22± 2.43	0.28±1.58	0.21±2.98	**Sparsity**
CAM/ Grad-CAM	0.25	0.25	0.17	0.36	0.11	**Fidelity**
	99.99 ± 0.39	100.00 ± 0.11	99.99± 0.11	100.0± 0.0	99.99± 0.29	**Contrastivity**
	2.82 ± 7.84	2.62 ± 8.07	6.26±7.83	9.36±7.67	4.86±8.74	**Sparsity**
Grad-CAM Avg	0.26	0.24	0.17	0.38	0.17	**Fidelity**
	58.41 ± 19.70	59.93 ± 20.47	41.06±19.05	29.22±14.04	44.03±23.7	**Contrastivity**
	0.57 ± 7.07	0.63 ± 7.30	0.01±0.07	0.0±0.0	0.01±0.11	**Sparsity**
EB	0.12	0.06	0.18	0.38	0.19	**Fidelity**
	29.68 ± 25.45	68.35 ± 36.35	50.87±18.76	60.29±15.40	49.06±22.59	**Contrastivity**
	60.92 ± 23.28	64.31 ± 22.48	40.35±22.11	51.4±13.97	30.12±23.04	**Sparsity**
c-EB	0.07	0.04	0.18	0.35	0.12	**Fidelity**
	81.46 ± 21.69	73.41 ± 37.76	96.97± 5.68	97.04±5.12	97.23±9.3	**Contrastivity**
	59.35 ± 23.13	76.31 ± 25.15	40.54±21.69	53.01±13.95	31.31±22.91	**Sparsity**

Table 2: Measures of fidelity, contrastivity, and sparsity for each method. The best performing method (on average) for each metric is highlighted in green (higher values are better).

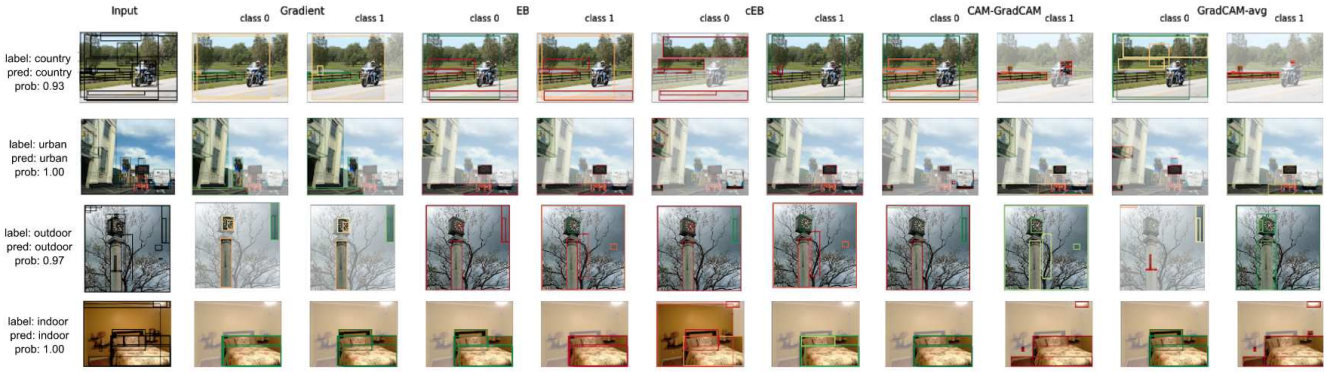

Figure 3: Selected explanation results for each Visual Genome dataset. All sample bounding boxes are shown in the Input column. Bounding boxes are colored according to their saliency value (red:low, green:high). Top 5 objects are shown for each method to reduce clutter. "Class 0" is either country or outdoor. "Class 1" is either urban or indoor. Contrastivity can be seen by comparing regions across classes for each method, e.g., the gradient method tends to highlight the same features for both classes, whereas CAM-GradCAM tends to highlight different objects. Best viewed on a computer screen.

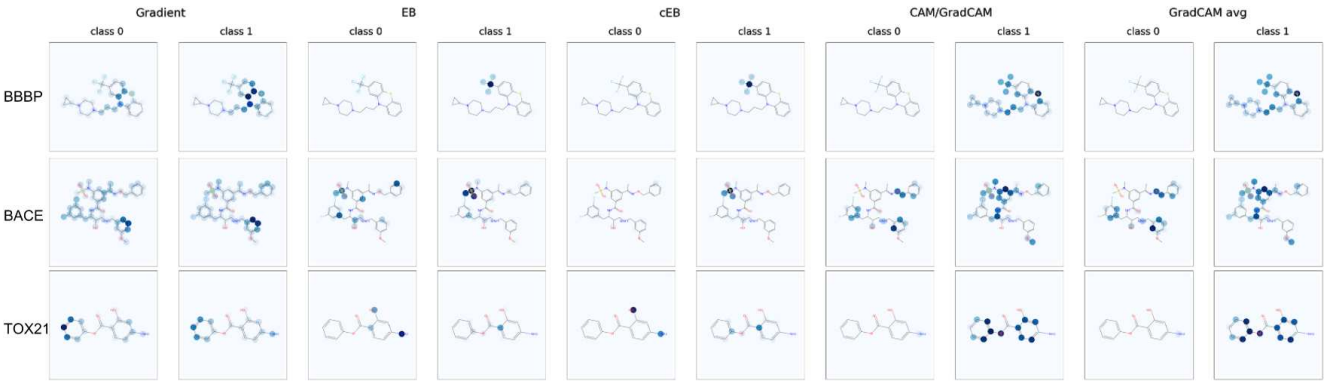

Figure 4: Selected explanation results for each molecule dataset. Each sample is a true positive. A darker blue color indicates a higher relevance for a given class. Sparsity can seen, e.g. in EB, by noting that fewer atoms are highlighted than other methods. Best viewed on a computer screen.

References

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1

[2] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014. 2

[3] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic. Signal recovery on graphs: Variation minimization. *IEEE Trans. Signal Processing*, 63(17):4609–4624, 2015. 2

[4] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević. Discrete signal processing on graphs: Sampling theory. *IEEE transactions on signal processing*, 63(24):6510–6523, 2015. 2

[5] F. Chollet. keras. https://github.com/fchollet/keras, 2015. 6

[6] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 1, 2

[7] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015. 1, 2

[8] P. Ertl. An algorithm to identify functional groups in organic molecules. *J Cheminform*, 9(1):36, Jun 2017. 7

[9] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 2017. 1

[10] N. Haider. Functionality pattern matching as an efficient complementary structure/reaction search tool: an open-source approach. *Molecules*, 15(8):5079–5092, Jul 2010. 7

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[12] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 1

[14] Z. Huang, C. Wan, T. Probst, and L. Van Gool. Deep learning on lie groups for skeleton-based action recognition. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1243–1252. IEEE computer Society, 2017. 2

[15] J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2

[16] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016. 2

[17] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim. The (un) reliability of saliency methods. *arXiv preprint arXiv:1711.00867*, 2017. 3

[18] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Advances in neural information processing systems*, 2017. 1, 2, 4

[20] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016. 5

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[22] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. E. Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *FE@ NIPS*, pages 196–212, 2015. 1

[23] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 1

[24] A. Mayr, G. Klambauer, T. Unterthiner, and S. Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016. 6

[25] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3, 2017. 1, 2

[26] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim. Image to image translation for domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018. 1

[27] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017. 1

[28] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2017. 3

[29] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018. 5

[30] K. Schütt, P. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 991–1001, 2017. 2

[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017. 1, 2, 5

[32] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 1, 2

[33] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 2

[34] J. Wu, B. Zhou, D. Peck, S. Hsieh, V. Dialani, L. Mackey, and G. Patterson. Deepminer: Discovering interpretable representations for mammogram classification and explanation. *arXiv preprint arXiv:1805.12323*, 2018. 2

[35] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 1, 4, 6

[36] L. Yi, H. Su, X. Guo, and L. J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 6584–6592, 2017. 2

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. *arXiv preprint arXiv:1806.01973*, 2018. 1

[38] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 1

[39] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. In *European Conference on Computer Vision*, pages 543–559. Springer, 2016. 1, 2, 3, 5

[40] Q. Zhang, Y. Nian Wu, and S.-C. Zhu. Interpretable convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 2

[41] Q.-s. Zhang and S.-C. Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018. 1, 2

[42] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016. 1, 2, 4