GCN-LRP explanation: exploring latent attention of graph convolutional networks

Jinlong Hu*, Tenghui Li, Shoubin Dong
Guangdong Key Laboratory of Communication & Computer Network, School of Computer Science and Engineering
South China University of China, Guangzhou, China
jlhu@scut.edu.cn

***Abstract*—Graph convolutional networks (GCNs) have been successfully applied to many graph data on various learning tasks such as node classification. However, there is limited understanding of the internal logic and decision patterns of GCNs. In this paper, we propose a layer-wise relevance propagation based explanation method for GCNs, namely GCN-LRP, to explore the latent pattern of GCNs. Then, we use three well-known citation network data sets and synthetic graph data sets for node classification tasks with GCN-LRP explanation, and experimentally identify latent attentions when GCNs aggregates information from the node and its neighboring nodes: (i) GCNs pay more attention to the classified node when comparing with its neighbors; (ii) GCNs do not pay attention to all the neighboring nodes equally, and a few neighboring nodes received more attention than others. Moreover, we further theoretically analyze and find that: (i) the latent attentions come from the recursively aggregating of GCNs; (ii) the neighboring nodes, which share enough neighbors with classified node, would receive more attention than other neighbors; (iii) the latent attention could hardly be changed by model training. We also discuss the advantage and limitations of GCNs introduced by the latent attentions, and implications of our findings for graph data learning with GCNs.**

I. INTRODUCTION

Tackling learning tasks on graph-structured data usually need extracting representations from nodes' features and graph structure. For this purpose, graph neural networks (GNNs) [1-8] compute new representations for each node by aggregating its neighbors' feature vectors or representations. Inspired by convolutional neural networks (CNNs) successfully used in computer vision [9-10], researchers began to extend convolution operation onto graphs as an aggregation method. Most works were based on spectral graph convolutional neural networks (GCNNs) [11-12], and graph convolutional networks (GCNs) was introduced in [3] which applied some simplifications to the basic frameworks and improved model's scalability and classification performance successfully. GCNs have been improved and applied to various application scenes [13-14].

To explain how the input relevant to the decisions of deep neural networks, various explaining method have been proposed for deep networks [15-18]. For example, Class Activation Mapping (CAM) [16], Excitation Back-propagation (EB) [17] and gradient-weighted CAM (Gard-CAM) [18] have been proposed for understanding how the input data relevant to model's decisions. Pope et al. summarize the explainability method for CNNs [19]. For graph neural networks, CAM, EB and Gard-CAM have been adapted to explain graph convolutional neural networks [20]. Besides the explaining works on the trained model, there are also some research on explaining the model training and generalization [21-26]. All these works provide insights to understand and improve the neural network models.

To figure out how the prediction values generated from original input values, layer-wise relevance propagation (LRP) was introduced in [27], which brings the explainability about how input data support the prediction in the trained model. Comparing with other explaining methods [16-18], LRP gives a more detailed description of propagating the prediction backward in the neural network, and the outputs of LRP can be delivered to the neural network for producing almost the same prediction value as its inputs. To explain the model, LRP are something different for Euclidean data (e.g. image data) and non-Euclidean data [28-29] (e.g. graph data). For image data, we can use the outputs of LRP to observe and confirm pixel contributions. Since images are grid-structured data in Euclidean space, the explaining outputs of LPR are only relevant to the grid value and learned weights. For graph data, the graph structure will also influence the output value of the explaining methods, so we could not use original LRP method to explain GCNs directly.

In this paper, we firstly propose a new layer-wise relevance propagation based explaining method for GCNs, namely GCN-LRP. Then, we explore the latent attentions of graph convolutional networks using GCN-LRP explanation in three well-known citation network data sets and a synthetic graph data set. Moreover, we further theoretically analyze where the latent attention comes from and why this attention can hardly be changed by label information of data and model training, and we also discuss the advantage and limitations of GCNs introduced by the latent attentions. Our main contributions are summarized as follows.

- We propose a new LRP based explanation method (GCN-LRP) for GCNs. It could explain the GCN model on feature level with the contributions of each feature from nodes on local group or category group; and explain the GCN model on node level with the contributions of each node.
- With the GCN-LRP explanation, we experimentally identify latent attentions of GCNs: (i) GCNs pay

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

more attention to the classified node when comparing with its neighbors; (ii) GCNs do not pay equal attention to all the neighboring nodes, and a few neighboring nodes received more attention than others.

- With further theoretical analysis, we find that: (i) the latent attentions come from the recursively aggregating of GCNs; (ii) the neighboring nodes, which share enough neighbors with classified node, would receive more attention than other neighbors; (iii) the latent attention could hardly be changed by model training.
- Our findings indicate GCNs would be more powerful when the characteristics of the learning tasks and data fit the latent attentions, and conversely, GCNs would be less powerful.

The paper is structured as follows. In Section 2, we review the LRP method and the GCN model, then propose the GCN-LRP method for GCNs explaining. In Section 3, we describe our experiments on citation network data sets and the synthetic graph data sets with GCN-LRP explanation, and our findings about the latent attention pattern of GCNs. In Section 4, we further analyze and discuss where the attention pattern from and why it can hardly be changed by label information of data and training procession. We conclude the paper in Section 5.

II. METHODS

A. Explaining neural networks with LRP

Layer-wise relevance propagation (LRP) [27] calculate each input neuron's contribution to the activation in neural networks, and these contributions are propagated back layer-wise as relevance scores. Because the layer-wise propagation relies on model structure and trained weights, LRP method explain how trained model treats all related input neurons when it makes decision.

For the typical neural network layer without bias, the input vector of l layer denote to be Z^l, and LRP method usually define that the relevance score R_j^l of neuron h_j^l as:

$$R_j^l = \sum_k \frac{a_j^l w_{jk}^l}{\sum_{0,j} a_j^l w_{jk}^l} R_k^{l+1}, A^l = \{a_j^l | j = 0, \dots, m\}, W^l = \{w_{jk}^l | j = 0, \dots, m; k = 0, \dots, n\} \quad (1)$$

Where Σ_k sums over the neurons related to neuron h_j^l in layer $l+1$ and $\Sigma_{0,j}$ sums over the neurons related to neurons $\{h_i^{l+1}\}$ in layer l. And LRP is subject to a conservation property, where what has been received by a neuron must be redistributed to the lower layer in equal amount. The layer-wise conservation property which can be verify is $\sum_j R_j^l = \sum_k R_k^{l+1}$, and by extension the global conservation property $\sum_i R_i^l = f(x)$.

There are some enhancements [30] of general LRP propagation rule (1), where one of the enhancements is adding a small positive term ϵ in the denominator:

$$R_j^l = \sum \frac{a_j^l w_{jk}^l}{\epsilon + \sum_{0,j} a_j^l w_{jk}^l} R_k^{l+1} \quad (2)$$

The role of ϵ is to absorb some relevance when the contributions to the activation of neuron k are weak or contradictory [31]. We can control ϵ be larger to achieve explanations being sparser in terms of input features and less noisy.

Another enhancement is favoring the effect of positive contributions over negative contributions:

$$R_j^l = \sum \frac{a_j^l \cdot (\sum w_{jk}^l + \gamma w_{jk}^{l+})}{\sum_{0,j} a_j^l \cdot (\sum w_{jk}^l + \gamma w_{jk}^{l+})} R_k^{l+1} \quad (3)$$

The parameter γ controls by how much positive contribution are favored. As γ increase, negative contributions start to disappear, this helps to deliver more stable explanations. The original idea of treat positive and negative contribution in any asymmetric manner was proposed in [27]. And we can consider the generic rule:

$$R_j^l = \sum \frac{a_j^l \rho(w_{jk}^l)}{\epsilon + \sum_{0,j} a_j^l \rho(w_{jk}^l)} R_k^{l+1} \quad (4)$$

So, LRP produces relevance score of any input feature for the decision of trained model.

B. Graph convolutional networks

Let a graph G = (V, E, A). V, E, A are vertex set, edge set and adjacency matrix of graph G, Assume N=|V| be the number of nodes in graph G, we have attributes $X \in R^{N \times d}$ and adjacency matrix $A \in R^{N \times N}$. In addition, we can calculate the degree matrix for this graph as $D_{ii} = \sum_j A_{ij}$.

Following the work in [3], each graph convolution layer constructs embedding for each node in graph according to $X^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X^l W^l)$, where $X^l \in R^{N \times d_l}$ is the embedding at the l layer and $X^0 = X$; $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix with added self-connections and W^l is trainable weights. The activation function $\sigma(.)$ is typically set to be the ReLU.

C. Explaining graph convolutional networks with LRP

In this subsection, we describe the GCN-LRP explanation method. We could treat the process of layer relevance propagation on GCN as inverse process of forward reasoning through graph convolutional network. In the forward reasoning, first, each node i calculates the weighted sum of all neighboring nodes' representations, the weight for each neighboring node j equals the value locate in the row i and column j of the normalized graph Laplacian L. Second, each node transforms its representation by multiplying W. So, as the inverse process, we assign relevance to components of node's feature vector firstly, then we distribute the node's relevance vector which be calculated in first step to all its neighboring node.

If we treat $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X$ as a whole part $\tilde{X}$, then we have $X^{l+1} = \sigma(Z^{l+1}), Z^{l+1} = \tilde{X}^l W^l$. It is just the same as typical neural network layer structure, so the first step is propagating relevance scores only in node's feature filed just like the normal LRP method:

$$\tilde{R}_j^l = \sum_k \frac{\tilde{x}_j^l w_{jk}^l}{\sum_{0,j} \tilde{x}_j^l w_{jk}^l} R_k^{l+1}, \ \tilde{X}^l = \{\tilde{x}_j^l | j = 0, \dots, m\}, \widetilde{W} = \{\widetilde{w}_{jk}^l | j = 0, \dots, m; k = 0, \dots, n\} \quad (5)$$

Next, feature relevance scores of the classified node shall be propagated to its all neighboring nodes on the graph.

According to the normalized graph Laplacian $L = \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$ and $\tilde{X} = LX$, the propagating formula could be defined as follow:

$$Rv_i^l = \sum_j \frac{X_k^l L}{\sum_{0,k} X_k^l L} \widetilde{Rv}_j^l, Rv_i^l = \{R_j^l | j = 0, .., n\}, X^l = \{x_j^l | j = 0, \dots, n\} \quad (6)$$

The multiplication between $\sum_j \frac{X_k^l L}{\sum_{0,k} X_k^l L}$ and $\widetilde{Rv}_j^l$ is the multiplication between the element in vectors, and the division between $X_k^l L$ and $\sum_{0,k} X_k^l L$ is division between the element in vectors.

The GCN-LRP method to explain node classification tasks of GCNs is described in Algorithm 1. In Algorithm 1, we use (5) to calculate preliminary relevance scores of node's features from Line 10 to 13. And we use (6) to calculate final relevance scores about this node's all neighboring nodes features from Line 16 to the end, where we first find all neighboring nodes through L, then propagate the relevance scores to all neighboring nodes in the graph. The output of Algorithm 1 is the relevance scores matrix R_0 which includes relevance scores of features in all related nodes.

In the spirit of open science and to promote reproducibility, we make our source codes publicly available on our project website (https://github.com/largeapp/gcn-lrp).

Algorithm 1: GCN-LRP for node classification tasks of GCNs.

Input: Activations $A = (X, a_1, \dots, a_l)$, wights $W = (w_1, \dots, w_l)$, bias $B = (b_1, \dots, b_l)$, normalized graph Laplacian L, classified node index k
Output: The relevance scores matrix R_0

```
1:  LA ← [LX, LA_1, ..., LA_{l-1}]
2:  Rs ← [None] * L + [A_l]
3:  nodes = [k]
4:  for i = 1,2,..., l do
5:      // calculate relevance scores for every layer
6:      R ← zeros(A[−i − 1].shape)
7:      nodes_next ← set([])
8:      for node in nodes do
9:          // use (5) to calculate preliminary relevance scores.
10:         z ← matmul(LA[−i][node,:]) + B[−i]
11:         s ← Rs[−i][node,:]/z
12:         c ← matmul(s, W[−i].T)
13:         r ← LA[−i][node,:] * c
14:
15:         // use (6) to calculate final relevance scores.
16:         index_nonzero ←find all nonzero element's index in L[node,:]
17:         nodes_next ← node_next + set(inde_nonzero)
18:
19:         R_node ← zeros(A[−i − 1].shape)
20:         L_vec ← L[node,:]
21:         vecs ← [A[−i − 1][j,:] * L_vec[j] for j in index_nonzero]
22:         sum_vec ← sum(vecs, 0)
23:         ratio_vecs ← [vec/sum_vec for vec in vecs]
24:         for k in index_nonzero do
25:             R_node[index_nonzero[k]] ← r * ratio_vecs[k]
26:         end
27:         R ← R + R_node
28:     end
29:     Rs[−i − 1] ← R
30:     nodes ← nodes_next
31: end
32: R_0 = R[0]
```

III. EXPERIMENTS

In this section, we apply our GCN-LRP method to explain GCN model on the citation network data sets, then analyze the output of GCN-LRP explanation in feature level and node level. And we construct synthetic graph data sets and design some learning tasks for further experiments.

A. Experiments on citation network data sets

1) Datasets: We experimented with three well-known citation networks data sets [31]: Citeseer, Cora and Pubmed, and their sparse feature vectors are generated from a synthetic list by using bag-of-words model, which represent whether the list words appear in a paper or not. Even the original citation links are directed, we still treat the citation links as undirected edges. See [3, 31] for more details about these data sets.

2) Experiment setup: We train a GCN model for semi-supervised classification on citation networks data sets and use GCN-LRP to explain the trained GCN model.

a) GCNs setup: We consider a two-layer GCN using in [3]. The middle layer size of this two-layer GCN are 16, and the middle layers activation function is ReLU and the output layer followed by a softmax classifier. Models are trained for 100 epochs using ADAM optimizer with learning rate 0.01. The models were implemented in Tensorflow, see [3] for more details about the model parameter. We split nodes to training, validation and test set with 1:3:6 in the experiments.

b) GCN-LRP setup: The LRP propagation rule for middle layer is defined by (2) and for output layer is defined by (3). For propagation between nodes, we define the rule as (6). The top layer relevance scores are set to the top-layer activation, which we multiply by a label indicator [30].

3) Explanation experiments on feature level: GCN aggregates the feature vectors of the neighbors to represent the central node, and we use GCN-LRP method to compute relevance scores (i.e. contributions) of all the related nodes' features for each node classification task.

a) Feature contributions in local group: A local group is defined by a group of the nodes including the classified node and its neighboring nodes. We calculate the feature relevance scores for all nodes in each local group, then sum the scores for each feature in the same local group. We define a feature contribution vector for each local group (namely LG-vector), whose elements are the sum scores for each feature.

b) Feature contributions in category group: A category group is defined by a group of nodes, which are in a same category, i.e. these nodes have a same label. We extract the most important features for nodes in each category group**:** (i) we calculate LG-vector for each node in the same category group; (ii) we calculate the mean value and standard deviation of all elements in each LG-vector, and we select all the features whose scores larger than the mean value added one standard deviation as the important-feature for the classified node in same category group; (iii) for every feature, we count the frequency when the feature is the important-feature for each node, and then select these features whose frequency larger than the mean value of all the frequency as the category-special features.

c) Effectiveness of GCN-LRP explanation: We validate effectiveness of GCN-LRP by occluding the salient features captured by explanation method. We occlude the features to observe the changing of classification accuracy in classifying nodes for each category. The occlusion is implemented by using zero-mask, i.e., setting the selected features to zero while keeping the rest constant.

The impact of category-special feature occlusion is shown in Figure 1. The feature occlusion leads to the classification accuracy of corresponding category an obviously fall, and the accuracy of corresponding category reaches nearly zero when the rate of occlusion reaches 40 percent. The classification accuracy of other categories is not impact by the occlusion. The detail results of the feature occlusion in Cora are shown in Table 1. The baseline is the classification accuracy for each category with occlusion, and the following row contains the accuracy when we occlude the features of corresponding category.

The results show that the category-special features captured by GCN-LRP are fidelity, i.e. these features are different enough between categories, therefore the occlusion of these features only affect the corresponding category. And the features captured by GCN-LRP also are complete enough, i.e. the occlusion of these features can lead the classification accuracy of corresponding category to drop to near zero.

4) Explanation experiments on node level

In the node level analysis, we define the norm of all the feature relevance scores of a node as the node contribution value. We transform the node contribution value to node contribution ratio (CR), through dividing the node contribution value by the sum of all the contribution value from nodes in local group.

a) Comparing contribution ratio of classified node with other nodes: We calculate the median, mean, standard deviation of CR from all nodes in each local group, and calculate CR of classified node for every node classification task. We take average ratio over all classification tasks.

The results are shown in Table 2, and the CR of classified node is always larger than the mean contribution ratio of all nodes in its local group, and it has over 72% chance to be the largest CR of nodes in all local groups from the three citation network data sets.

b) Contribution ratio of neighboring nodes: We divided whole contribution interval (between the maximum value and the minimum value of contributions from all neighboring nodes) into eight subintervals, and we define the ratio of node number (RNN) for each subinterval as RNN=Ni/Nall, where Ni is the number of the nodes whose contribution value is in the subinterval, and Nall is the number of all neighboring nodes.

We also define the ratio of node contribution (*RNC*) for each subinterval as $RNC=C_i/C_{all}$, where C_i is the total contribution of the nodes whose contribution value is in the subinterval, and C_{all} is the total contribution of all neighboring nodes.

We take average RNN and RNC over all classification tasks in the Figure 2. The results show that besides classified

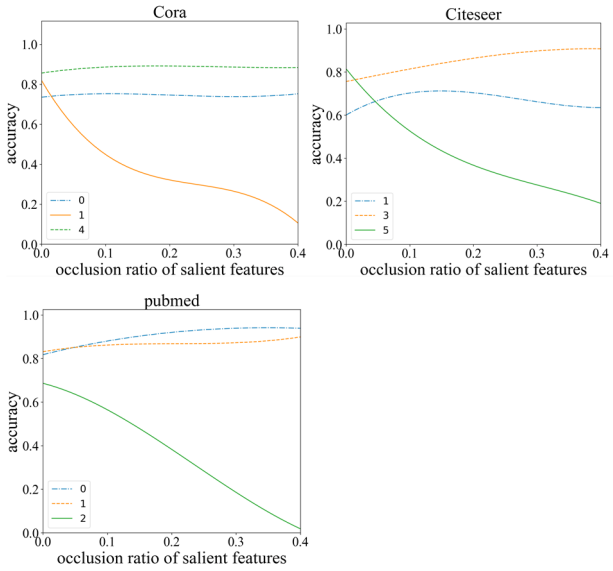

Fig. 1. For Cora data set, the salient features of category 1 are occluded, compared for classification accuracy changing of category 1, category 0 and 4. For Citeseer data set, the salient features of category 5 are occluded, compared classification accuracy changing of category 5, category 1 and 3. For Pubmed data set, the salient features of category 2 are occluded, compared for classification accuracy changing of category 2, category 0 and 1.

TABLE I. OCCLUSION FOR EACH CATEGORY IN CORA

	Classification Accuracy						
	Category. 1	***Category .2***	***Category. 3***	***Category. 4***	***Category. 5***	***Category. 6***	***Category. 7***
Baseline	0.79	0.89	0.92	0.76	0.86	0.76	0.77
Occluding-F1	0.18	0.84	0.99	0.80	0.90	0.78	0.73
Occluding-F2	0.74	0.13	0.85	0.71	0.86	0.78	0.73
Occluding-F3	0.77	0.86	0.15	0.53	0.91	0.76	0.61
Occluding-F4	0.81	0.88	0.83	0.18	0.67	0.83	0.73
Occluding-F5	0.61	0.82	0.97	0.55	0.25	0.77	0.93
Occluding-F6	0.67	0.59	0.96	0.77	0.94	0.12	0.67
Occluding-F7	0.47	0.85	0.94	0.77	0.96	0.69	0.22

TABLE II. NODE CONTRIBUTION RATIO

	Median CR	**Mean CR**	**Std CR**	**CR of Classified node (ratio of being the largest one)**
Cora	0.0843	0.0951	0.0459	0.1844 (72.1%)
Citeseer	0.1867	0.1965	0.0607	0.2865 (72.2%)
Pubmed	0.0384	0.0473	0.0294	0.1405 (84.5%)

node, there are only a few nodes have large contribution ratio, and many other nodes have very small contribution ratio.

c) Findings and discussion on latent attention: Based on the above two contribution ratio experiments, we could find out: when GCNs aggregate node information in node

classification tasks, there are latent attention behavior patterns of GCNs as follows.

- For the information aggregations of nodes, the classified node's own information is attached more importance by GCNs, i.e. GCNs usually pay more attention to the classified node than the neighboring nodes. And the attention to the classified node has a high probability (above 72% in the experiments) to be the maximum from all related nodes.
- In each node's information aggregation, beside itself, neighboring nodes also are not treated equally by GCNs, where only a few neighboring nodes are received more attentions, and other nodes have very limited contribution for the classification.

B. *Experiments on synthetic data*

In this subsection, we will validate these attention patterns are not generated from label information of data or training process, but from the latent mechanism of GCNs.

1) Data and learning task: To further exploit the latent attention of GCNs, we construct the synthetic graph data and corresponding classification tasks. The synthetic graph data set is constructed by randomly setting 3000 nodes with one binary feature (one or zero), and all edges are randomly generated between these nodes, and we adjust the generation probability to control the graph density in the experiments. Then we present two labeling methods for nodes and corresponding classification tasks. For each node, we consider its all one-hop and two-hop neighborhood when we labeling it.

- If the number of its neighbors' feature being *one* more than the number of being *zero*, we set node's label to be *label-one*, otherwise we set node's label to be *label-zero*. And this labeling method correspond the classification task *task-1*.
- We compute the sum of the neighborhood's feature value. If the sum is an odd number, we set the node's label to be *label-one*, otherwise we set node's label to be *label-zero*. And this labeling method correspond the classification task *task-2*.

For each classification task, we split nodes to training, validation and test sets as 1:3:6. Obviously, the *task-1* and *task-2* need the learning model (i) to treat all neighbor nodes equally when classify one node, because every neighborhood's contribution for the label are equal; (ii) to pay less attention to classified node, because classified node's feature not affect its label.

We also construct two additional labeling methods based on *task-1* for further experiments.

- We add the classified node's value to count the sum of number of *one* or *zero* in *task-1*, and for emphasizing classified node's impact, when its value being *one*, we count it as three times of being *one*. And this labeling method correspond the classification task *task-11*.
- For emphasizing few special nodes' impact, we select the node sharing the largest number of neighborhood with classified node, and count it as three times when its value being *one*. And this labeling method correspond the classification task *task-12*.

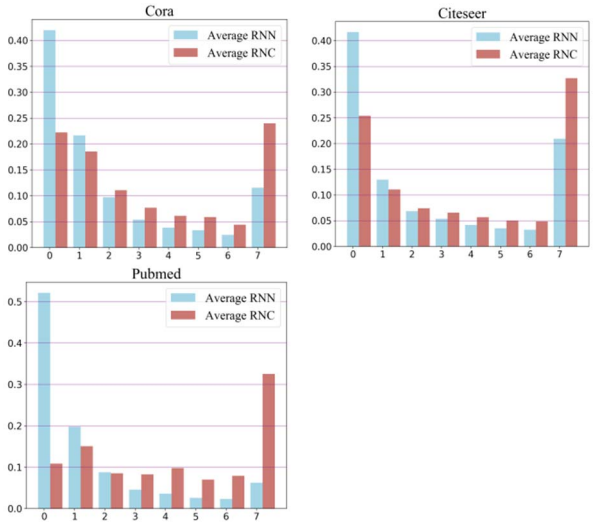

Fig. 2. For eight subintervals, left bar represents average RNN of each subinterval, and right bar represent average RNC of each subinterval.

TABLE III. COMPARISON ON CLASSIFICATION ACCURACY (IN PERCENT)

Method	task-1	task-2	task-11	task-12
Two-layer GCNs	53.5	54.3	93.9	91.0
Three-layer GCNs	53.5	50.0	93.9	91.0
FCNNs	100.0	100.0	100.0	100.0

TABLE IV. NODE CONTRIBUTION RATIO IN SYNTHETIC DATA

Task	median	mean	std	CR of Classified node (ratio of being the largest)
Task-1	0.1723	0.2122	0.1170	0.4400 (99.7%)
Task-2	0.1783	0.2190	0.1240	0.4508 (100%)

2) Experiment setup: We consider a two-layer GCN model whose receptive field just covers one-hop and two-hop neighborhood and a three-layer GCN model whose receptive field covers a little bigger than one-hop and two-hop neighborhood. The size of middle layer in two-layer GCN model is 16, and the size of middle layers in three-layer GCN model is 16 and 4 respectively. The middle layers activation and other general learning parameters are same to the experiments on citation network data sets. For further comparing, we construct a typical full-connected neural networks (FCNNs) model with one hidden layer including 16 hidden neurons. Since FCNNs model can't aggregate node's neighborhood, we use one-hop and two-hop neighborhood's features as the classified node's features.

3) Results and discussion: In Table 3, both two-layer GCNs and three-layer GCNs are failed to classify the nodes in *task-1* and *task-2*, while the FCNNs are successful to classify the nodes. Compared with *task-1*, GCN models greatly improve the accuracy of node classification in *task-*

11 and *task-12*, where the classified node itself or its special neighbors are emphasized in these tasks.

In Table 4, GCN models pay more attention to classified node than its neighborhood in *task-1* and *task-2*, and the classified node has high probability to be the most important one in nodes from local group.

As shown in Figure 3, RNN and RNC of each subinterval are different, and it means that GCN models could not treat the neighborhood nodes equally.

IV. Discussion

A. Where the latent attention comes from

In a layer of GCNs, any node's related row in normalized graph Laplacian drives it to aggregate its neighboring nodes' information. This process iterates through layer superposing, and it increases each node's receptive field to high-order neighbor. And this process also causes each node aggregating repeatedly its own information and some special neighbors'. For two-layer GCNs as an example, in the first layer, the classified node's information could be collected once by its all neighbor, and in the second layer, all neighbors return back this part of information to classified node through this node's aggregating. In this process, classified node's information could be magnified. And for the same reason, the nodes that share enough neighbors with classified one can be magnified obviously.

To simplify further theoretically analysis, we ignore the weights W^l and the activation function in each layer, only refer two times aggregation driven by normalized graph Laplacian, and we have:

$$f_{2i} = \frac{1}{\mathrm{d_i}} f_{1i} + \sum_j \frac{1}{\sqrt{d_i d_j}} f_{1j} \tag{7}$$

$$f_{3i} = \left(\frac{1}{d_i^2} + \sum_j \frac{1}{d_i d_j}\right) f_{1i} + \sum_j \left(\left(\frac{1}{d_i} + \frac{1}{d_j}\right) \frac{1}{\sqrt{d_i d_j}} + \frac{1}{d_j} \sum_k \frac{1}{\sqrt{d_i d_k}} \right) f_{1j} + \ldots \tag{8}$$

Where f_{1i}, f_{2i}, f_{3i} are the classified node's input features, first-layer aggregation features and second-layer aggregation features respectively. The f_{1j} is neighboring node's input features, k is the index of all shared neighboring nodes between classified node i and neighboring node j. The two-layer GCN consider both first-order proximity and second-order proximity when aggregates node's information. And d_i is the degree of node i or degree add one(if add self-connections when use GCN), so in the unweighted graph, classified node's information magnification scale $1/d_i^2 + \sum_j 1/d_i d_j$ is positive correlation with the number of its neighboring nodes, and this magnification scale is limited by the degree of its neighboring nodes. Neighboring nodes' information magnification scale $(1/d_i + 1/d_j)/\sqrt{d_i d_j} + \left(1/d_j\right) \sum_k 1/\sqrt{d_i d_k}$ is mainly positive correlation with the number of neighboring nodes which share neighbor between themselves and the classified node, and this magnification scale is limited by the degree of the nodes sharing neighbors.

With the superposing of layers, higher order neighbors are considered. When ignore the weights W^l and activation function, for n-layer GCN, the ratio of GCN's attention between classified node i and its neighboring node j is just the ratio between $L^n(i,i)$ and $L^n(i,j)$, where the L^n is the normalized Laplacian to the power of n.

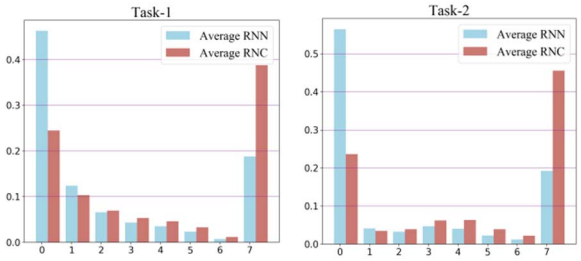

Fig. 3. Bar charts for the relations between the number and contribution of nodes in interval.

In summary, the aggregation method and preferences of GCNs cause latent attentions before graph data training, and these latent attentions are limited by the trained graph data structure (e.g. degree of the neighbors of classified node, or degree of nodes that sharing neighbors with classified node).

B. Why the latent attention can hardly be changed

Based on the above experiments and analysis, we could find that the nodes which are originally received more attention by GCNs (before training), are still important from the output of GCN-LRP explanation in trained model (after training), whatever the label information of data and tasks are fit enough to latent attentions (e.g. citation network experiments, and task-11 and task-12) or not (e.g. task-1 and task-2).

To simplify further analyze why the latent attention can hardly be changed, we consider the layer weights W^l of GCNs, and the weights of all layers can be considered as one transform weight matrix W. And then we define the question as: for any node classification task, what kinds of weights that we get after training can smooth initial latent attention between the classified node i and its neighboring node j. We define the smoothing as balancing the output attention, which means contribution ratio between classified node and its neighboring node is 1:1 or less. Considering contributions of the node as the L^2 norm of node's feature contribution vector, so the question can be translated to: for one classified node i and its neighboring node j, assuming their feature contribution vectors are X and Y, what kinds of weights can keep a high probability to transform the ratio x/y between the norm of X and Y is 1:1 or less. For example, in two-layer GCN, x is $1/d_i^2 + \sum_j 1/d_i d_j$, and y is $(1/d_i + 1/d_j)/\sqrt{d_i d_j} + \left(1/d_j\right) \sum_k 1/\sqrt{d_i d_k}$:

$$\frac{x}{y} = \frac{1/d_i^2 + \sum_j 1/d_i d_j}{(1/d_i + 1/d_j)/\sqrt{d_i d_j} + \left(1/d_j\right) \sum_k 1/\sqrt{d_i d_k}} \tag{9}$$

For $\in R^{n \times m}$, consider its singular value decomposition (SVD) $W = U \Sigma V^T$ and the left singular matrix $U = (X_1, X_2, \ldots, X_n)$. All X_k in $\{X_k | 1 \le k \le n\}$ are mutually orthogonal, and because of $U^T W = (X_1 W, X_2 W, \ldots, X_n W) = \Sigma V^T$, all $X_k W$ which in $\{X_k W | 1 \le k \le n\}$ are mutually orthogonal. We assume that with multiplying by matrix W, the norm scaling factors for basis vectors $\{X_k | 1 \le k \le n\}$ are $\{w_k | \|X_k W\|_m = w_k \cdot$

$\|X_k\|_n, 1 \le k \le n\}$. Any feature contribution vector can be described by standard basis as $X = a_1X_1 + a_2X_2 + \cdots + a_nX_n$, so we have its norm change ratio between XW and X:

$$\frac{\|XW\|_m}{\|X\|_n} = \sqrt{\frac{w_1^2a_1^2+w_2^2a_2^2+\cdots+w_n^2a_n^2}{a_1^2+a_2^2+\cdots+a_n^2}} \quad (10)$$

Assuming $Y = b_1X_1 + b_2X_2 + \cdots + b_nX_n$, and we have the ratio between YW and XW :

$$\frac{\|YW\|_m}{\|XW\|_m} = \sqrt{\frac{w_1^2b_1^2+w_2^2b_2^2+\cdots+w_n^2b_n^2}{w_1^2a_1^2+w_2^2a_2^2+\cdots+w_n^2a_n^2}} \quad (11)$$

For any W to smooth the initial attention unbalance, the corresponding $\{w_i|1 \le i \le n\}$ should make (11) be larger or equal to (9) when $A^2 = (a_1^2, a_2^2, \dots, a_n^2)$ is close to $B^2 = (b_1^2, b_2^2, \dots, b_n^2)$ (since neighboring nodes' features are similar). In other words, for the set $N = \{A^2, B^2, \dots\}$ in the nonnegative orthant of n dimensional space, the model need find a vector $w^2 = (w_1^2, w_2^2, \dots, w_n^2)$, which make inner product with some close vector pair in N and ensure the ratio $\sqrt{(w^2, B^2)/(w^2, A^2)}$, can be bigger than the ratio x/y as many as possible. It's obvious that if the neighboring node pairs distance's upper limit l be smaller or (9) be larger, finding the w^2 will be more difficult, i.e. changing node attention can be harder for GCN in training. Actually, the value of ratio (9) is one order higher than the distance between neighboring node pairs (e.g. in Cora data set, the mean value of (9) is 25.94 and the mean distance between neighboring node pairs is 0.30). It means the space for model learning a weight matrix which can smooth latent attention is very limited. Assuming all vectors in N evenly distributed on the unit hypersphere in nonnegative orthant of n dimensional space, and the included angle between A^2 and w^2 are α, and the included angle between B^2 and w^2 are β, then (11) can be $\sqrt{cos\beta/\cos\alpha}$, considering neighboring node pairs vector's included angle less than θ (e.g. in Cora data set, θ is 0.02 meanly, and the max θ is 0.25), so the root of ratio between A^2 and B^2 can be:

$$f(\beta) = \sqrt{\frac{\cos(\beta)}{\cos(\beta+\theta)}} \quad (12)$$

Take $\theta = 0.25$ for example, if w^2 locate in the middle of the orthant, β must be less than $\pi/4$, and $f(\beta)$ is less than 1.17, almost negligible to the mean value of ratio (9) in Cora. And following w^2 move approach to coordinate axis, the upper limit of β raises. And only when β can be larger than $5\pi/12$, $f(\beta)$ can approach the mean value of ratio (9) in Cora, and in means w^2 must be near coordinate axis, i.e. the space for W to be learned by training is very limited if model want to smooth the latent attention. Even if the w^2 is close to coordinate axis after training, only node pairs both have the included angle larger than $5\pi/12$ with w^2 can be smoothed latent attention. It means for smoothing initial latent attention, not only the weight W is hardly to be learned, but also the distribution requirements of nodes' feature vector are very strict.

C. *The advantage and limitations of GCNs introduced by latent attentions*

In this subsection, based on the findings of latent attentions, we discuss the advantage and limitations of GCNs in term of training data and learning tasks.

The latent attentions bring naturally unbalance of initial node attention, since usually only a few nodes are emphasized. In the dense graph where k is closed to j in (8) or deep GCN [14] which superpose enough layer, this initial node attention can be more balance, but it could bring the problem of being over-smoothing [32]. Therefore, when the label information of data and classification tasks fit the latent attention patterns, GCNs would be more powerful for training and generalization; when the tasks are different with the latent attentions of GCNs, or the tasks need model treat nodes more equally, e.g. task-1 and task-2, the latent attentions could bring difficulty for training and be a burden to generalization for GCNs.

For example, in two-layer GCNs, if the tasks need to aggregate more information from the neighboring node which shares less neighbor with classified nodes, GCNs would be not good at the tasks, because the neighboring node shares enough neighbor with classified node would be important for the information aggregation. What's more, since the latent attention can hardly be change by after training, the trained models are difficult to be better.

V. CONCLUSION

In this paper, a new LRP based explanation method (GCN-LRP) for GCNs was proposed, and then GCN-LRP was applied to citation network data sets and synthetic graph data sets to explore the latent attentions of GCNs. With the experiments, it was found that the classified node and a few special neighbors would receive more attention than other nodes. With further theoretical analysis, it was found that the latent attentions come from the recursively aggregating of GCNs and the attentions could hardly be changed by model training. Based on the findings of latent attentions, the advantage and limitations of GCNs were discussed from in term of training data and learning tasks. Moreover, our findings indicate GCNs would be more powerful when the characteristics of the learning tasks and data fit the latent attentions, and conversely, GCNs would be difficult for the learning tasks.

In the work presented here, the data sets are limited to the citation networks and the synthetic graph data--it would be important to further explore the latent attentions in other data sets. Future work should focus on further exploring and analyzing the internal patterns of graph neural networks taking advantage of our explanation methods and findings.

ACKNOWLEDGMENT

This work was supported in part by the Natural Science Foundation of Guangdong Province of China [grant #2018A030313309], the Innovation fund of introduced high-end scientific research institutions of Zhongshan [grant #2019AG031], and the Fundamental Research Funds for the Central Universities, SCUT [grant #2019KZ20].

REFERENCES

[1] W. L. Hamilton, R. Ying, and J. Leskovec. "Inductive representation learning on large graphs." In Advances in Neural Information Processing Systems (NIPS), pp. 1025–1035, 2017a.

[2] 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.

[3] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks." In International Conference on Learning Representations (ICLR), 2017.

[4] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. "Gated graph sequence neural networks." In International Conference on Learning Representations (ICLR), 2016.

[5] A. J.-P. Tixier, G. Nikolentzos, P. Meladianos, M.Vazirgiannis. "Classifying graphs as images with convolutional neural networks." arXiv preprint arXiv:1708.02218 (2017).

[6] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. "Graph attention networks." In International Conference on Learning Representations(ICLR), 2018.

[7] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. "Representation learning on graphs with jumping knowledge networks." In International Conference on Machine Learning (ICML), pp. 5453–5462, 2018.

[8] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung. "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs." In UAI, 2018.

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

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

[11] 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.

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

[13] S. Abu-El-Haija, A. Kapoor, B. Perozzi, J. Lee. "N-GCN: Multi-scale graph convolution for semi-supervised node classification." arXiv preprint arXiv:1802.08888 (2018).

[14] W.-L. Chiang, X. Liu, S. S Y. Li, S. Bengio, C.-J. Hsieh. "Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks." In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2019).

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

[16] 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.

[17] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, S. Sclaroff. "Top-down neural attention by excitation backprop." International Journal of Computer Vision 126.10 (2018): 1084-1102.

[18] 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.

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

[20] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann. "Explainability methods for graph convolutional neural networks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 10772–10781, 2019.

[21] ZJ. Xu. "Understanding training and generalization in deep learning by fourier analysis." arXiv preprint arXiv:1808.04295 (2018).

[22] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. "Understanding deep learning requires rethinking generalization." arXiv preprint arXiv:1611.03530 (2016).

[23] K. Xu, W. Hu, J. Leskovec, S. Jegelka. "How Powerful are Graph Neural Networks?" arXiv preprint arXiv:1810.00826 (2018).

[24] S. Verma, Z.-L. Zhang. "Stability and Generalization of Graph Convolutional Neural Networks." Accepted at The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2019).

[25] H. N. Mhaskar and T. Poggio. "Deep vs. shallow networks: An approximation theory perspective." Analysis and Applications 14, 06 (2016), pp.829–848.

[26] S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin. "Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization." Advances in Computational Mathematics 25, 1-3 (2006), 161–193.

[27] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. M ¨uller, and W. Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation." PLoS ONE, 10(7):e0130140, 2015.

[28] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains." IEEE Signal Processing Magazine 30, 3 (2013), 83–98.

[29] B. L. Douglas. "The weisfeiler-lehman method and graph isomorphism testing." arXiv preprint arXiv:1101.5211, 2011.

[30] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, W. Same. "Layer-wise relevance propagation: an overview." Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer, Cham, pp. 193-209, 2019.

[31] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher. "Collective classification in network data." AI magazine 29.3 (2008): 93-93.

[32] Q. Li, Z. Han, and X.-M. Wu. "Deeper insights into graph convolutional networks for semi-supervised learning." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.