SPECTRALLY SIMILAR GRAPH POOLING

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of learning compositional hierarchies of graphs. Even though structural characteristics of graphs can be learned by Graph Neural Networks (GNNs), it is difficult to find an overall compositional hierarchy using such flat operators. In this paper, we propose a new graph pooling algorithm, Spectrally Similar Graph Pooling (SSGPool), to learn hierarchical representations of graphs. The main idea of the proposed SSGPool algorithm is to learn a coarsening matrix which maps nodes from an original graph to a smaller number of nodes in a coarsened graph. The coarsening matrix is trained to coarsen the nodes based on their feature vectors while keeping the spectral characteristics of the original graph in the coarsened one. Experiments on various graph benchmarks show the advantage of our method compared to strong baselines. To further investigate the effectiveness of our proposed method, we evaluate our approach on a real-world problem, image retrieval with visual scene graphs. Quantitative and qualitative analyses on the retrieval problem confirm that the proposed method efficiently captures the hierarchical semantic structure of scene graphs.

1 INTRODUCTION

By virtue of the recent progress on graph neural networks (GNNs) (Gori et al., 2005; Scarselli et al., 2008; Bruna et al., 2013; Kipf & Welling, 2016; Gilmer et al., 2017; Veličković et al., 2018), various types of data including structural data can be dealt with using neural network algorithms. While conventional neural network algorithms, such as convolutional neural networks and recurrent neural networks, take regular structured inputs (images with grid pixel structure and sound signals with Markovian temporal dependencies), GNNs have been recently suggested as a method for extending the scope of the inputs to graphs having irregular structures, such as molecular data, knowledge graphs, social networks and visual scene graphs. Most GNNs attempt to implicitly reflect the structural information through node (graph) representations. In other words, GNNs assign feature vectors to each node and update the node features by transforming and aggregating information from the neighborhoods. Even though structural characteristics can be learned by applying these message passing steps repeatedly, it is difficult to find an overall compositional hierarchy using such flat operators.

Recent work has proposed using pooling methods such as CNNs in order to discover hierarchical structures between nodes in GNNs (Vinyals et al., 2015; Ying et al., 2018; Zhang et al., 2018; Lee et al., 2019; Gao & Ji, 2019; Diehl, 2019; Ma et al., 2019). These studies are divided into two categories depending on what information is mainly used for the pooling operator: structure-based approaches and feature-based approaches. Structure-based approaches learn node features with GNNs, however, the original graph is coarsened by deterministic graph clustering algorithms based on graph theory. Therefore, the resultant coarsened graph reflects the topology of the original graph, but the node features are not used during coarsening. Also the deterministic clustering methods are not end-to-end trainable. On the other hand, feature-based approaches learn to assign nodes in the original graph to the nodes in the coarsened graph based on the node feature vector. Even though these approaches can be trained in an end-to-end manner, it is hard to maintain the topology information of the original graph.

In this paper, we propose a new graph pooling method, Spectrally Similar Graph Pooling (SSGPool), which makes use of both node features and structural information between the nodes (Figure 1). The main idea of SSGPool is to learn a coarsening matrix which maps nodes from an original graph to a smaller number of nodes in a coarsened graph. The coarsening matrix is trained to

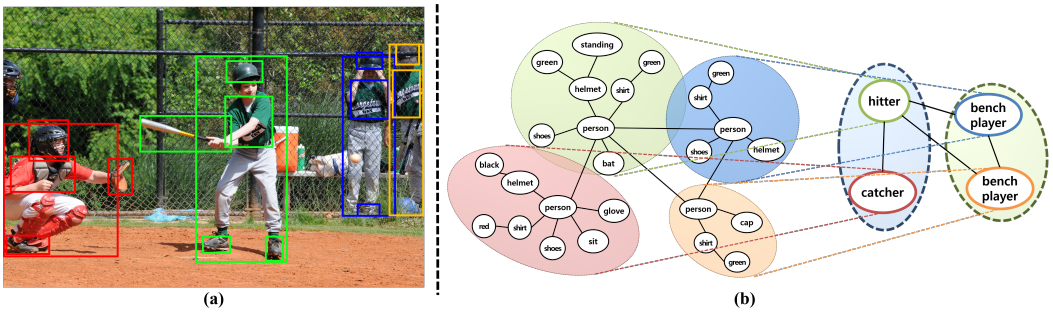

Figure 1: An illustrative example of compositional hierarchy in a visual scene graph. (a) is an original image and (b) is a hierarchical structure of visual scene graph for the image.

coarsen the nodes based on correlations between their feature vectors while maintaining the topology information using spectral characteristics of the original graph. To utilize the node feature vectors, SSGPool basically builds upon conventional GNN algorithms. In addition, structural similarities between two different sized graphs are defined in order to be used as a regularizer during training. By having structural similarities act as a regularizer, SGGPool binds nodes having similar feature vectors while keeping the spectral characteristics of the original graphs in an end-to-end manner.

Experiments on various graph benchmarks show the advantage of our method compared to strong baselines. To further investigate the effectiveness of our proposed method, we evaluate our approach on a real-world problem, image retrieval with visual scene graphs. Quantitative and qualitative analyses on the retrieval problem confirm that the proposed method efficiently captures the hierarchical semantic structures of scene graphs.

The remainder of the paper is organized as follows. In Section 2, we review related work about the graph pooling algorithms. Next, we introduce notations about the graphs, GNN algorithms and spectral similarity between graphs as preliminaries. After that, the proposed SSGPool method is explained in detail and experimental results on various datasets, comparing our proposed algorithm with other well-known graph pooling algorithms are presented.

2 Related Work

Pooling operations in graph neural networks (GNNs) can scale down the size of inputs and enlarge the receptive fields, thus giving rise to better generalization and performance. In this section, we review several recent methods for graph pooling coupled with GNNs. Graph pooling methods can be grouped into the following two categories: structure-based pooling and feature-based pooling.

2.1 Structure-based Pooling

Including earlier works of neural networks on graph, several proposed GNNs perform pooling with existing graph clustering algorithm. These methods learn the representations of graphs in 2-steps: First these pooling methods build hierarchical structures using a graph clustering algorithm. Next, they learn embeddings of nodes in each layer based on GNN modules. Bruna et al. (2013) built a hierarchy of the graph with agglomerative clustering. Defferrard et al. (2016) and Fey et al. (2018) used the Graclus algorithm (Dhillon et al., 2007) which computes graph clustering without eigenvectors. Simonovsky & Komodakis (2017) constructed the graph hierarchies through a combined use of spectral polarity and Kron reduction. More recently, Ma et al. (2019) proposed EigenPool, which used spectral graph clustering methods to produce a coarsened graph. These methods leverage topological information from graphs in order to produce coarsened graph. However these methods do not use node features which have useful information for learning representations of graphs. Furthermore, as the existing graph clustering algorithms are not differentiable, they are incapable of learning in an end-to-end fashion.

2.2 FEATURE-BASED POOLING

In contrast to structure-based pooling, several end-to-end trainable pooling methods are proposed. Ying et al. (2018) proposed a differentiable graph pooling module (DiffPool) to softly assign nodes to a set of clusters using neural networks, forming fully connected coarsened graphs through a dense cluster assignment matrix. Gao & Ji (2019) and Lee et al. (2019) devised a top-K node selection-based pooling method (gPool and SAGPool) to form an induced subgraph for the next layer. Although it is efficient, this method loses the completeness of the graph structure information. In addition, Vinyals et al. (2015) proposed Set2Set, the global pooling operation by aggregating information through RNNs. Zhang et al. (2018) proposed SortPool which pools graphs according to the feature map values that are sorted in descending order. Diehl (2019) designed a pooling operation by contracting the edges (EdgePool). The contracting scores are calculated by features from the two incident nodes. These approaches learn hierarchical structures from node features with differentiable parameters. However, they tend not to reflect the topology information of the graph for pooling.

3 PRELIMINARIES

3.1 GRAPH NOTATIONS

A graph G is denoted as a pair $(\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_1, ..., v_N\}$ the set of nodes (vertices), and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ the set of edges. Each node v_i is associated with a feature vector $x_i \in \mathbb{R}^f$. To make notation more compact, the set of node feature vectors of graph G is denoted as a matrix $X = [x_1, x_2, ..., x_N]^\top \in \mathbb{R}^{N \times f}$. Also, a graph has a N-by-N weighted adjacency matrix A where $A_{i,j}$ represents the weight of the edge between v_i and v_j and a degree matrix D, a diagonal matrix which contains information about the degree of each node — that is, the sum of edge weights attached to each node. As usual, we denote the combinatorial Laplacian L of graph G with $L = D - A$ and let λ_k and μ_k be the k-th (smallest) eigenvalue and corresponding eigenvector of L respectively.

3.2 GRAPH NEURAL NETWORKS

Due to an ever increasing interest in combining deep learning and structured approaches, various graph-based neural networks have been proposed over the years. Based on spectral graph theory (Chung & Graham, 1997), approaches which convert graphs to the spectral domain and apply convolution kernels of the graphs have been proposed (Bruna et al., 2013; Henaff et al., 2015; Kipf & Welling, 2016). Gilmer et al. (2017) suggested the message passing framework, which encompasses a number of previous neural models for graphs under a differentiable message passing interpretation. Xu et al. (2018) analyzed the representation power of various GNN architectures and proposed Graph Isomorphism Networks (GIN), where representational power is equal to the power of the Weisfeiler-Lehman test.

In this paper, we use a simple form of a message passing function similar to GIN.

$$\mathrm{M}(A, X) = (X + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} X) W_m \tag{1}$$

where $W_m \in \mathbb{R}^{f \times f}$. After that, we define a single GNNs layer as follows:

$$\mathrm{GNN}(A, X) = \left[\sigma\left(\mathrm{M}_2\left(A, \sigma\left(\mathrm{M}_1\left(A, X\right)\right)\right)\right); \sigma\left(\mathrm{M}_1\left(A, X\right)\right)\right] W_g \tag{2}$$

where M_1 and M_2 are message passing layer, σ is an activation function, $[X; Y]$ denotes row-wise concatenation of two matrix and $W_g \in \mathbb{R}^{2f \times f'}$ is a learnable parameter for $\mathrm{GNN}(A, X)$. In the rest of the paper, we use $\mathrm{GNN}(A, X)$ in Equation equation 2 as a base GNNs module.[1]

3.3 SPECTRAL SIMILARITY BETWEEN GRAPHS

Spectral graph theory has been considered as a powerful way to describe structural characteristics of graphs. Therefore, structural similarity between two graphs can be clearly defined by comparing the spectral properties of graphs.

[1] It is just for fair comparison with stable performances for all models. It is a non-critical choice and it can be substituted by any GNN architectures.

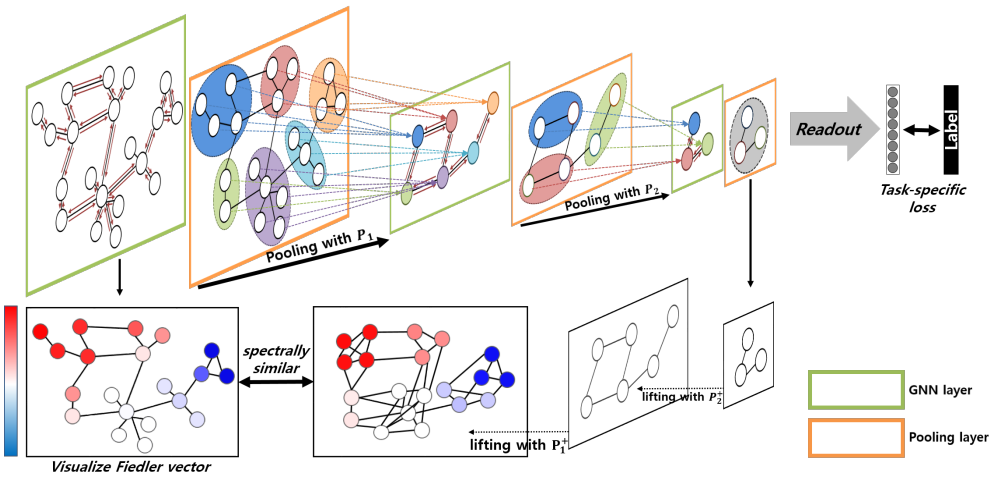

Figure 2: The architecture of SSGPool layer combined with graph neural networks. The SSGPool learns coarsening matrices P to minimize task-specific loss while retaining spectral similarity. To represent the spectral similarity, we use the Fiedler vector of graph Laplacian.

For two graphs having the same number of nodes, Spielman & Srivastava (2011) and Spielman & Teng (2011) proposed *spectral similarity* to determine how closely a graph G_s approximates a G:

$$\forall f \in \mathbb{R}^N, \quad (1-\epsilon) f^\top L f \leq f^\top L_s f \leq (1+\epsilon) f^\top L f \tag{3}$$

where L is a Laplacian matrix of a graph G. If the equation holds, we can say that G_s is an ϵ-spectral approximation of G.

For the graph coarsening problem which has different number of nodes between the original graphs and coarsened graphs, Loukas & Vandergheynst (2018) generalized it by restricting to first K-eigenspace: the *restricted spectral similarity (RSS)*. If there is a mapping matrix $P \in \mathbb{R}^{N \times n}$ between original vertex set $\mathcal{V} = \{v_1, ..., v_N\}$ and the coarsened vertex set $\mathcal{V}_c = \{v'_1, ..., v'_n\}$, then RSS is defined as follows:

Restricted Spectral Similarity (RSS). *Suppose that there exist an integer K and positive constant ϵ_k, such that for every $k \leq K$,*

$$(1-\epsilon_k) u_k^\top L u_k \leq u_k^\top \tilde{L} u_k \leq (1+\epsilon_k) u_k^\top L u_k, \quad \tilde{L} = P^{\mp} L_c P^{+}, \quad L_c = P^\top L P \tag{4}$$

where u_k is k-th eigenvector of L, P^+ and $P^\mp$ are pseudo-inverse of P and its transpose, and $\tilde{L}$ is Laplacian matrix of *lifted* (reverse of coarsening) graph of G_c from $\mathbb{R}^n$ back to $\mathbb{R}^N$. Then, the G_C is said to satisfy the restricted spectral similarity property with the RSS constants $\{\epsilon_k\}_{k=1}^K$.

4 Spectrally Similar Graph Pooling

We suggest a new graph pooling algorithm which learns coarsening matrix to construct adjacency matrix and node feature matrix of upper layers while keeping spectral characteristics of original graphs. The main idea is to keep the spectral information by maximizing the similarity between the Fiedler vector of original graphs and its coarsened ones. As two vectors are on different dimensional spaces, the vector of the coarsened graph is lifted back to the original space using the inverse of the coarsening matrix. In order to make the whole process end-to-end trainable, we define the coarsening matrix and derive the easy inversion of the coarsening matrix. Figure 2 shows the architecture of proposed method.

4.1 Graph Coarsening

The coarsening can be expressed with a surjective map (i.e., many-to-one) $\varphi : \mathcal{V}_N \to \mathcal{V}_n$ between the original vertex set $\mathcal{V}_N$ and the smaller vertex set $\mathcal{V}_n$. Then, graph coarsening can be defined via a coarsening matrix:

Definition 1 (Coarsening matrix). *Matrix $P \in \{0,1\}^{N\times n}$ is a coarsening matrix with regard to graph G if and only if it satisfies the condition that it is a surjective mapping of the vertex set, meaning that if $P(i,r) = 1$ then $P(i,r') = 0$ for every $r' \neq r$.*

Similar to Loukas (2019), the expensive pseudo-inverse computation for P can be substituted by simple transposition and re-scaling:

Proposition 1 (Easy inversion). *The pseudo-inverse of a coarsening matrix P is given by $P^+ = Q^{-2}P^\top$, where $Q \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $Q(r,r) = ||P(:,r)||_2$.*

4.2 Pooling with Coarsening Matrix

Suppose we have the learned coarsening matrix at l-th layer, $P_l \in \mathbb{R}^{N_l \times N_{l+1}}$. With P_l, SSGPool layer coarsens the graph, generating a new coarsened adjacency matrix A_{l+1} and a new node feature matrix X_{l+1}.

Most previous coarsening based pooling approaches such as Ying et al. (2018) and Ma et al. (2019) used a quadratic form of the adjacency matrix to obtain new coarsened adjacency matrix, $A_{l+1} = P_l^\top A_l P_l$. Instead, we use the Laplacian matrix L_l to obtain a new coarsened adjacency matrix A_{l+1}:

$$L_{l+1} = P_l^\top L_l P_l, \quad A_{l+1} = D_{l+1} - L_{l+1} \tag{5}$$

where D_{l+1} is a degree matrix obtained by leaving only diagonal terms of L_{l+1}.

Utilizing L_l instead of A_l has two noteworthy benefits. First, the obtained coarsened adjacency matrix is not diagonal-dominant: the coarsened graph obtained from the quadratic form of A has significantly stronger self-loops than any other connections, and these self-loops might hamper the message passing of GNNs. Second, our coarsening is consistent with regard to the Laplacian form: the Laplacian matrix of the coarsened graph retains spectral properties as is desired, e.g., the nullspace of L is preserved both by coarsening and lifting because $P_l \mathbf{1}_{N_{l+1}} = \mathbf{1}_{N_{l+1}}$ and $P_l^+ \mathbf{1}_{N_l} = \mathbf{1}_{N_l}$.

Further, the new node feature matrix of the next layer X_{l+1} is obtained as follows:

$$\begin{aligned} Z_l &= \text{GNN}_{l,embed}(A_l, X_l) \\ X_{l+1} &= P^+_{l,\text{soft}} Z_l \end{aligned} \tag{6}$$

where P_{soft} is softmax output of P, which will be covered in the next section. It is worthwhile to note that while most of previous methods use the form of transpose of P_{soft} so that features of upper nodes are obtained by sum of the original nodes (sum pooling), we use pseudoinverse of P_{soft} to weighted average the node features (average pooling) to get features of supernodes. As the number of nodes in each cluster can be vary, our method can stabilize the learning.

4.3 Learning the Coarsening Matrix

We describe how SSGPool generates the coarsening matrix at the l-th layer, $P_l \in \mathbb{R}^{N_l \times N_{l+1}}$. For convenience, we drop the notation of layer l and denote $\boldsymbol{p}_i = P(i,:)$. According to Definition 1, $\boldsymbol{p}_i$ can be defined as a categorical random variable with probabilities $\pi_{i1}, \pi_{i2}, ..., \pi_{in}$, where n is the number of nodes in the coarsened graph.

It is straightforward to sample from $\boldsymbol{p}_i$, but we cannot backpropagate gradients though the sampling since the variables are discrete. A recently popular approach to handle this difficulty is to sample from a continuous approximation of the discrete distribution (Maddison et al., 2016; Jang et al., 2017), and use the reparameterization trick to get (biased) gradients from this approximation. In this work, we simply borrow the gradient trick of Straight-Through Gumbel-Softmax estimator (Jang et al., 2017) to ensure end-to-end training. The probability π is estimated via the GNN module followed by softmax function:

$$\Pi = P_{\text{soft}} = \text{softmax}\left(\text{GNN}_{pool}(A, X)\right) \tag{7}$$

Finally, the $\boldsymbol{p}_i$ can be drawn by one-hot of the argmax on the softmax output:

$$\boldsymbol{p}_i = \texttt{one\_hot}\left(\arg\max_j [\pi_{ij}]\right) \tag{8}$$

Although the original ST-Gumbel trick utilizes samples drawn from $g \sim \text{Gumbel}(0,1)$ to give stochasticity, we drop this sampling procedure and choose the max j only with the probability π.

4.4 SPECTRAL SIMILARITY OF GRAPHS AS REGULARIZATION

In this section, we propose the spectral regularizer for a graph pooling, which enforces coarsening matrices to keep coarsened graph spectrally similar to the original graph. To start with, the relationship between the original graph and the final coarsened graph is expressed in compact form:

$$L_f = P_* L_0 P_*^\top, \quad \tilde{L}_0 = P_*^+ L_f P_*^{\mp} \tag{9}$$

where L_f and L_0 are Laplacian matrices of the final coarsened graph and the original graph, $P_* = P_f \cdots P_0$ and $P_*^+ = P_0^+ \cdots P_f^+$. By virtue of Proposition 1, the pseudo-inverse of P_l can be calculated in linear time.

In spectral graph theory, the second smallest eigenvector of graph Laplacian, also known as Fiedler vector, entails the overall topology information of graphs, as it is the function that maps adjacent nodes with similar values: The larger difference between values of nodes has the farther topological distance between nodes is.

The Fiedler vector u of the original graph can be coarsened and lifted given a coarsening matrix P_*:

$$u_c = P_*^+ u, \quad \tilde{u} = P_* u_c \tag{10}$$

where $\tilde{u}$ is a vector that has been sequentially coarsened and lifted from Fiedler vector vector u given the matrix P_*. Then, the $\tilde{u}$ is the best approximation of u given P_*, because the $P_* P_*^+$ is the projection matrix with a smaller rank (See the proof of Proposition 1). Therefore, as the distance between $\tilde{u}$ and u gets closer, the original graph and coarsened graph become more similar to each other in terms of global structure. Finally, we propose the spectral regularizer term based on cosine similarity:

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Task}} + \lambda \cdot \left(1 - \frac{u^\top \tilde{u}}{|u| \cdot |\tilde{u}|}\right) \tag{11}$$

where $\mathcal{L}_{\text{Task}}$ is task-specific loss term and λ is a hyperparameter for the regularization term.

Connection to Restricted Spectral Similarity (RSS). Followed by Loukas (2019), The RSS can be re-formulated through the following induced semi-norm:

$$||u||_L = \sqrt{u^\top L u}, \quad ||u_c||_{L_c} = \sqrt{u_c^\top L_c u_c}, \quad \text{where} \quad u_c = P_*^+ u$$
$$(1-\epsilon)||u_c||_L \leq ||u_c||_{L_c} \leq (1+\epsilon)||u||_L \tag{12}$$

Then, we can obtain an upper bound of difference between semi-norms of the original graph and the coarsened graph with a triangular inequality.

$$\frac{||u - \tilde{u}||_L}{||u||_L} \geq \frac{|\,||u||_L - ||u_c||_{L_c}|}{||u||_L}, \quad \text{where} \quad \tilde{u} = P_* u_c \tag{13}$$

Therefore, reducing the distance between u and $\tilde{u}$ with our regularization term makes the original graph and coarsened graph to be spectrally similar.

5 EXPERIMENTS

In this section, we highlight the advantages of SSGPool compared to other competitive graph pooling algorithms with various graph benchmark datasets. Also, we apply our method to a real-world problem, image retrieval with visual scene graphs. For the experiments, we use five competitive baselines recently proposed for differentiable graph pooling. All the experimental details including baseline models, benchmarks and implementation details are in Appendix B.

5.1 GRAPH CLASSIFICATIONS TASK WITH GRAPH BENCHMARK DATASETS

We evaluate SSGPool on a variety of graph datasets from benchmarks commonly used for graph classification tasks. To examine the general ability of our model, four datasets are selected according to their amount of data and graph size: MUTAG (Debnath et al., 1991), ENZYME (Borgwardt et al., 2005), PROTEINS (Feragen et al., 2013) and NCI1 (Shervashidze et al., 2011).

Table 1: Average accuracy and standard deviation for graph benchmarks are presented. The DiffPool* denotes DiffPool with additional losses originally proposed in Ying et al. (2018). Also, SSGPool-NoReg indicates the SSGPool without regularization. We highlight the best results (**bold**) and second best results (blue).

Model	MUTAG	ENZYME	PROTEINS	NCI1
GNN w/o Pooling	.746±.007	.301±.023	.726±.007	.733±.005
SortPool (Zhang et al., 2018)	.832±.016	.277±.020	.730±.012	.734±.011
gPool (Gao & Ji, 2019)	.732±.018	.303±.019	.734±.006	.721±.004
SAGPool (Lee et al., 2019)	.803±.015	.326±.028	.730±.006	.738±.009
EdgePool (Diehl et al., 2019)	.770±.033	.329±.025	.731±.004	.751±.006
DiffPool* (Ying et al., 2018)	**.853**±.019	.283±.043	**.756**±.009	.743±.009
SSGPool-NoReg	.846±.015	.369±.021	.745±.006	.752±.005
SSGPool	.852±.009	**.382**±.012	.750±.005	**.753**±.010

Table 1 shows overall results for graph benchmarks compared to other state-of-the-art graph pooling methods. The average and standard deviation are obtained from 10 times of 10-fold cross validations test. First of all, we highlight that the proposed regularization term significantly improves performance across all datasets. This implies that preserving global structures while simultaneously pooling graphs has a substantial impact on graph representation learning.

We observed that, for ENZYME and NCI datasets, the SSGPool showed best performance. Even though the SSGPool achieves second best performance in MUTAG and PROTEINS datasets, it shows very competitive results compared to other methods. Also, it is worthwhile to note that the selected four datasets have distinct statistics in terms of the number of data and graph size. As a result, all comparative models show considerably different performance depending on the datasets. For example, The DiffPool shows best performance at MUTAG and PROTEINS but for the ENZYME and NCI1, it achieves degraded scores. However, the proposed method consistently showed good performance across all datasets. We also report the results for benchmarks with varying hyperparameters (e.g., the number of pooling layer, pooling ratio and existence of regularizer) in Appendix C.

5.2 Image Retrieval Task with Visual Scene Graph

To see more intuitive interpretation of the pooling, we apply SSGPool to perform image retrieval via visual scene graph matching. A visual scene graph, initially proposed in Johnson et al. (2015), represents contents of an image in the form of a graph consisting of three kinds of components: objects, their attributes, and relationships between two objects.

Table 2: The results of image retrieval in terms of NDCG. Higher the NDCG score is, better the performance.

	NDCG					
Model	5	10	20	30	40	50
ResNet152	.720	.728	.742	.756	.771	.786
GNNs	.785	.799	.820	.836	.845	.862
SAGPool	.789	.803	.824	.839	.852	.865
DiffPool	.790	.805	.825	.840	.853	.865
SSGPool	**.796**	**.810**	**.830**	**.844**	**.857**	**.869**

Visual scene graphs can be used to build an image-to-image retrieval system (Gordo & Larlus, 2017), which returns a list of images sorted by relevance with respect to an image query. In an image retrieval system based on a visual scene graph, the relevance measure is defined as a degree of matching between visual scene graphs. The matching between two visual scene graphs can be evaluated by computing their cosine similarity between embedded visual scene graphs, either annotated by a human or algorithmically generated, into a fixed-length vector.

To train and evaluate the image retrieval system, we need a ground truth measure of image relevance. Following prior work (Gordo & Larlus, 2017) which demonstrates that the similarity between image captions is highly correlated to the human rating of image relevance, we utilize caption similarity as a proxy metric during our experiment. We use S-BERT (Reimers & Gurevych, 2019), a transformer pretrained to generate sentence embeddings, to compute the similarity between captions. A proxy relevance measure between two images is obtained by first computing S-BERT representations of the captions and then obtaining the cosine similarity between them. With the proxy relevance score

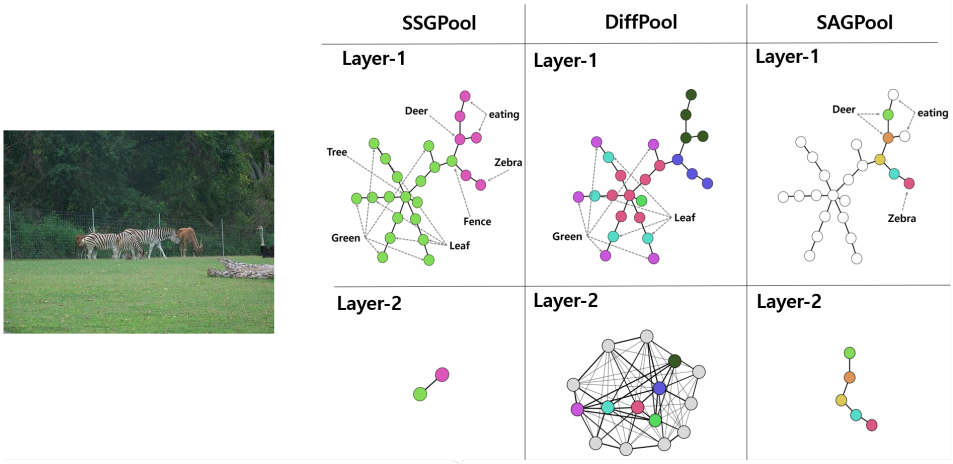

Figure 3: Left : An original image corresponding to the scene graphs on the right. Right : Pooling results on each graph in each layer. Same color of nodes are meant to be mapped to the same coarsened node in the pooled layer. Since DiffPool coarsens the graph with soft assignment matrix, we selected a top-1 coarsened node for each original node for visualization. The grey colored nodes in layer-2 are left-over coarsened nodes that were not chosen as top-1 by any original nodes. Some significant node labels are specified to demonstrate different properties between the methods.

defined, Normalized Discounted Cumulative Gain (NDCG) is used to measure the performance of retrieval.

The proxy relevance score also provides supervision for learning graph representation. In every iteration, a batch of training image pairs (and corresponding visual scene graph pairs) are sampled, and the squared error between the cosine similarity of embeddings in each pair and their proxy relevance score is minimized. To obtain both captions and scene-graphs for images, we use 48,220 images which belongs to both MS COCO dataset (Lin et al., 2014) and Visual Genome (VG) dataset (Krishna et al., 2017). Following the Stanford split (Xu et al., 2017), we manually split the VG-COCO dataset with 36,601 train, 1,000 validation and 5,000 test images. We use ResNet152 (Simonyan & Zisserman, 2014), GNNs without pooling, DiffPool and SAGPool are chosen as comparative baselines. Table 2 shows the performance on the image retrieval task. Among the overall models, the SSGPool achieves the best results over all NDCG scores.

To compare the learned hierarchical structure among the graph pooling methods, we visualize the coarsening results of each model (Figure 3). As shown in the first column, SSGPool coarsens the graph by reflecting the structural information well. Due to this characteristic, the trees and their attributes (leaf-green) are coarsened to a single node, and deer eating grass and zebra are coarsened to another node. Furthermore, it can be seen that our method successfully maintains the overall topological structure of the original graph in the upper layer. In the case of DiffPool taking the coarsening form like our method, however, nodes with similar features tend to be coarsened together. Also, as DiffPool has a dense coarsening matrix, the upper layer graph cannot reflect the original graph structure and has the form of a fully connected graph. Lastly, the SAGPool constitutes hierarchies by selecting important nodes. We can see that it selects important nodes (e.g., eating, deer, zebra) but loses considerable amounts of other peripheral information. Additionally, SAGPool's upper layer graph loses structural information from the original graph due to it is masking out all other nodes not selected. We attach more examples of qualitative results in Appendix D.

6 Conclusions

In this paper, we proposed the end-to-end graph pooling method, Spectrally Similar Graph Pooling. In contrast to previous work, our method learns compositional hierarchies while preserving the global structure of the graph. The proposed method shows competitive results not only in graph benchmarks datasets, but in real-world problem such as image retrieval with visual scene graphs. We also show that our proposed method learns meaningful hierarchical structures.

REFERENCES

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. American Mathematical Soc., 1997.

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pp. 3844–3852, 2016.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11): 1944–1957, 2007.

Frederik Diehl. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990*, 2019.

Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Towards graph pooling by edge contraction. In *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*, 2019.

Aasa Feragen, Niklas Kasenburg, Jens Petersen, Marleen de Bruijne, and Karsten Borgwardt. Scalable kernels for graphs with continuous attributes. In *Advances in neural information processing systems*, pp. 216–224, 2013.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 869–877, 2018.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *International Conference on Machine Learning*, pp. 2083–2092, 2019.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Albert Gordo and Diane Larlus. Beyond instance-level image retrieval: Leveraging captions to learn a global visual representation for semantic retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6589–6598, 2017.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3668–3678, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International Conference on Machine Learning*, pp. 3734–3743, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pp. 3243–3252, 2018.

Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 723–731, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5410–5419, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.