Unsupervised Hierarchical Graph Representation Learning by Mutual Information Maximization

Fei Ding
School of Computing, Clemson University
Clemson, USA
feid@clemson.edu

Xiaohong Zhang
Department of Chemical and Biomolecular Engineering,
Clemson University
Clemson, USA
xiaohoz@clemson.edu

Justin Sybrandt
School of Computing, Clemson University
Clemson, USA
jsybran@clemson.edu

Ilya Safro
School of Computing, Clemson University
Clemson, USA
isafro@clemson.edu

ABSTRACT

Graph representation learning based on graph neural networks (GNNs) can greatly improve the performance of downstream tasks, such as node and graph classification. However, the general GNN models do not aggregate node information in a hierarchical manner, and can miss key higher-order structural features of many graphs. The hierarchical aggregation also enables the graph representations to be explainable. In addition, supervised graph representation learning requires labeled data, which is expensive and error-prone. To address these issues, we present an unsupervised graph representation learning method, Unsupervised Hierarchical Graph Representation (UHGR), which can generate hierarchical representations of graphs. Our method focuses on maximizing mutual information between "local" and high-level "global" representations, which enables us to learn the node embeddings and graph embeddings without any labeled data. To demonstrate the effectiveness of the proposed method, we perform the node and graph classification using the learned node and graph embeddings. The results show that the proposed method achieves comparable results to state-of-the-art supervised methods on several benchmarks. In addition, our visualization of hierarchical representations indicates that our method can capture meaningful and interpretable clusters. **Reproducibility:** Our code and experimental data are available at this link[1].

CCS CONCEPTS

• **Information systems** → Data mining; • **Mathematics of computing** → Graph algorithms; • **Computing methodologies** → Unsupervised learning.

KEYWORDS

graph neural networks, representation learning, unsupervised learning, hierarchical representation, mutual information

1 INTRODUCTION

Graph representation learning has been used in many domains that are related to graph-structured data, including bioinformatics [9], chemoinformatics [18, 29], social networks [8] and cybersecurity [40]. There are two important tasks in graph analysis, *i.e.*, label predictions on nodes and graphs. For instance, in the study of chemical molecules, researchers apply graph classification [22, 23, 43] to help discover chemical properties of new molecule by predicting labels of the molecule, where a molecule can be represented as a graph with the atom represented as nodes and chemical bond represented as edges.

Graph neural networks (GNNs) are applied to graph-based data to improve prediction performance due to their ability to learn high-level features by propagating, transforming, and aggregating neighborhood information across edges [11, 14]. There are various neighborhood aggregation methods to capture the structures and attributes of graphs, including the average aggregation [19], generalized aggregation [14] and attention-based aggregation [39]. However, these techniques sometimes miss key structural features for large, sparse, and noisy real-world graphs. In these cases, the most valuable information is often contain in several small subgraphs, which conventional aggregations methods often struggle to capture.

To solve this problem, Lee *et al.* [22] present the Graph Attention Model (GAM), which focuses on small parts of graphs in order to predict the labels of the entire graphs. In order to improve embedding quality, the GAM model also integrates global information from various parts of the graph via different random sets of nodes. This suggests that local and global information are both important in graph representation learning. In the analysis of real-world graphs, it is necessary to gather information from individual nodes and edges as well as the subgraphs of graph that represent discriminative patterns. Recently, Ying *et al.* [43] proposed a graph pooling module, DiffPool, to generate hierarchical representations of graphs for the purpose of graph classification. This mechanism allows GNNs to encode the local and global structural information to obtain the final graph representation. Although the above methods perform well in the graph classification task, they are task-specific and focus on supervised learning. These methods depend highly on vast quantities of labeled graph data, which is often costly and error-prone in the real world. To address this problem, Veličković *et al.* [39] applies mutual information maximization to learn node representations of graph-structured inputs without using labelled data, and demonstrates competitive performance to supervised learning on several node classification benchmarks.

[1] https://github.com/ifding/uhgr

Inspired by this work, we propose a novel unsupervised learning method, Unsupervised Hierarchical Graph Representation (UHGR), to learn hierarchical graph representations based on mutual information maximization, which includes node embeddings and graph embeddings. We summarize the main contributions as follows:

- We propose an unsupervised hierarchical graph representation learning method to capture the local and global structural information of arbitrary sized graphs, which does not depend on any task-specific information (e.g., class labels). This method is generic enough to be used in various scenarios such as node embedding and graph embedding.
- We demonstrate that the graph representations from the proposed model can achieve comparable node and graph classification performance to supervised baseline methods on real-world data sets.
- The proposed method can learn meaningful and interpretable clusters across different levels of coarseness based on the structural information of graphs, as demonstrated through our visualizations.

The remainder of this paper is organized as follows: section 2 illustrates our proposed method; the discussions of experimental results are provided in section 3; section 4 reviews the related work; finally, we discuss the conclusions in section 5.

2 PROPOSED METHOD

Inspired by the recent success of unsupervised learning based upon mutual information maximization [16, 39], we propose a novel unsupervised embedding framework, UHGR, to capture structural information and learn a hierarchical graph representation. This method is based on the maximization of mutual information between "local" features from neighbors of one node and high-level "global" features from the entire graph, which enables us to learn both node and graph representations.The proposed method utilizes the unsupervised learning method to aggregate structural information to generate hierarchical representations. This unsupervised method makes the graph representations feasible for various downstream tasks, such as node and graph classification. Meanwhile, our method overcomes the shortcomings of previous studies that do not integrate different structural information of graphs well. To evaluate our method, we apply the learned representations on the node and graph classification tasks, and compare the classification results with several baseline methods.

2.1 Preliminaries

The undirected graph $G = (X, A)$ is comprised of n nodes, each with f features. Here, $X \in \mathbb{R}^{(n \times f)}$ where the original node features $\vec{x_i}$ is read directly from files and represented by row i of X. Furthermore, the adjacency matrix $A \in \{0, 1\}^{(n \times n)}$ contains a nonzero entry A_{ij} to indicate an edge between nodes i and j. The goal of this work is to create different levels of low-rank encodings of G, which we accomplish by training an encoder to cluster local parts of the graph and create more coarsened graphs, eventually output the final representation of the original graph. Each coarsened graph has its own node features and an adjacency matrix that are trainable. In order to train the encoder module, we apply a hierarchical approach where G is repeatedly coarsened from $G_1 = (H_1, A_1), \ldots, G_L(H_l, A_l)$ and H represents the learned representations. The H_1 and A_1 are from the original graph, and the H_l is the final graph representation of the original graph. Following this scheme, the number of nodes in the successively coarsened graphs is non-increasing. Because $H_i \in \mathbb{R}^{n_i \times f_i}$ represents the node embeddings of level$i$, if $i < j$, then $n_i \leq n_j$. The feature vector corresponding to the coarse nodes is determined by a separate hierarchical level, $G_{i-1} \rightarrow \mathbb{R}^{f_i}$, which learns node embeddings of level i from the previous level $i - 1$ of coarseness.

This paper uses graph neural networks (GNNs) to create representations of the graphs at different levels, which is able to capture hierarchical structures and generate flexible graph embeddings. A key component of the proposed method is how to cluster partial parts of the graph and generate more coarsened graphs based on the output of GNNs without any labels. In the following parts, we outline the different modules of UHGR and illustrate how to learn hierarchical graph representations based on mutual information maximization.

2.2 Encoder module

The hierarchical encoder mainly depends on message-passing function $\mathcal{M}$. The message-passing function $\mathcal{M}$ is used to iteratively compute node representations from their neighborhood's features [43]:

$$H^{(k)} = \mathcal{M}(A, H^{(k-1)}), \tag{1}$$

where $H^{(k)}$ are the node embeddings of the k-th step from message-passing function $\mathcal{M}$, which depends on the adjacency matrix A and the previous node embeddings $H^{(k-1)}$. At the initial step (k=1), $H^{(0)}$ is initialized by the original node features X. After K iterations, the module outputs the final node embeddings $Z = H^{(k)}$. The message-passing function $\mathcal{M}$ can be implemented by different types of GNNs. In this work, we consider two general GNNs: Graph Convolutional Networks (GCNs) [19] and Graph Attention Networks (GATs) [38].

Graph Convolutional Networks. GCNs implement $\mathcal{M}$ using the following rule:

$$H^{(k)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(k-1)}W^{(k-1)}), \tag{2}$$

where $\hat{A} = A + I$ is the adjacency matrix with self-loops and $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ is the corresponding degree matrix. For the nonlinearity σ, we apply the parametric ReLU function [15], and $W^{(k-1)}$ is a trainable weight matrix.

Graph Attention Networks. GATs leverage self-attentional layers to set learnable weights to measure the importance of neighborhoods when aggregating feature information from node's neighbors. When computing new feature representation for a central node, each neighborhood receives a different weight by measuring the relation between its feature vector and the central node's vector. Node i and its neighborhood node j have the following relations:

$$e_{ij} = a(W\vec{x_i}, W\vec{x_j}), \tag{3}$$

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}), \tag{4}$$

where e_{ij} is the attention coefficients and a represents a single-layer feed-forward neural network to perform self-attention on the nodes. The shared weight matrix W is used for every node to perform linear transformation. α_{ij} indicates the importance of

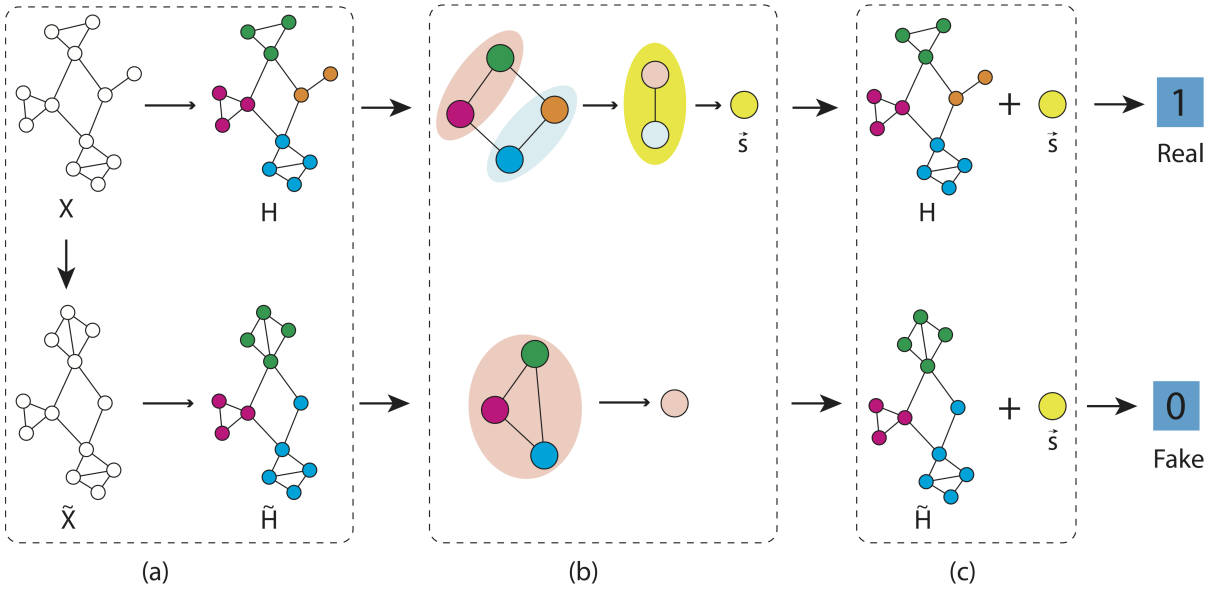

Figure 1: The architecture of the Unsupervised Hierarchical Graph Representation (UHGR) model. The left module (a) is an Encoder that creates the node representations H by exploiting the node feature X and the adjacency matrix A. The middle module (b) utilizes hierarchical graph pooling to create the graph summary $\vec{s}$. The right module (c) is a Discriminator trained to discriminate if a pair of H and $\vec{s}$ is generated from the same graph or not.

node j's features to node i after normalizing e_{ij} during the feature aggregation process.

2.3 Graph pooling module

To assign nodes to clusters at each hierarchical layer, we apply DIFFPOOL [43] to create node embeddings and adjacency matrix for next coarsened layer (i+1) from layer i.

$$Z^{(i)} = \text{GNN}(A^{(i)}, H^{(i)}), \tag{5}$$

$$(A^{(i+1)}, H^{(i+1)}) = \text{DIFFPOOL}(A^{(i)}, Z^{(i)}). \tag{6}$$

The graph pooling module takes the adjacency matrix $A^{(i)}$ and the features of the nodes or cluster nodes at layer i as the input of the GNN module to get the new embedding matrices $Z^{(i)}$ of nodes or cluster nodes. Then the DIFFPOOL module takes the node embedding matrices $Z^{(i)}$ and the adjacency matrix $A^{(i)}$ to generate a coarsened adjacency matrix $A^{(i+1)}$ and new embeddings $H^{(i+1)}$ for each of the nodes or cluster nodes in this coarsened graph. Then, the new coarsened graphs are fed to the GNN module to generate a coarser version of the input graph. This whole process is repeated several times until the final graph representation is generated, which contains only one general node or cluster node. Compared to other hierarchical representation learning methods, our model learns a hierarchical representation strategy automatically, which doesn't depend on the specific task and can be trained end-to-end. Generally, this unsupervised manner embeds the original graph to a coarser one by grouping the similar subgraphs together.

2.4 Discriminator module

Similar to Deep InfoMax [16, 39], we introduce a discriminator module to help training the Encoder module and Graph pooling module, which enables our model to output the satisfied representations. The discriminator module trains the encoder to maximize the mutual information between a high-level graph representation and local features of the graphs and it is able to capture the unique graph representation for each graph individually. The local features are also included in the learned node embeddings, which represents the hierarchy of the original graphs. In this context, the final output representation of hierarchical learning is the graph-level summary representation $\vec{s}$, and the local graph features are from the node embeddings of the original graph $H = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_n\}$. Therefore, our hierarchical model can be written by the following equation:

$$\vec{s} = \mathcal{R}(\text{GNN}(A, H)), \tag{7}$$

where A represents an adjacency matrix of the original graph, and $\mathcal{R} : \mathbb{R}^{n \times f} \rightarrow \mathbb{R}^{f'}$ is used to obtain a hierarchical graph-level representation. GNN module can be any node embedding module such as GCN and GAT. The readout function $\mathcal{R}$ utilizes the unsupervised hierarchical process to summarize the graph-level vector $\vec{s}$.

For the objective function, we follow the same loss function as DGI [39], which computes the standard binary cross-entropy between graph samples from the joint and the product of marginals:

$$\mathcal{L} = \frac{1}{n+m}\Big(\sum_{i=1}^{n} \mathbb{E}_{(H,A)}[\log \mathcal{D}(\vec{h}_i, \vec{s})] + \sum_{j=1}^{m} \mathbb{E}_{(\tilde{H},\tilde{A})}[\log(1 - \mathcal{D}(\vec{\tilde{h}}_i, \vec{s}))]\Big), \tag{8}$$

Table 1: Data set summary used in node classification task

Data set	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3

Table 2: Data set summary used in graph classification task

Data set	Graphs	Classes	Avg.# Nodes	Avg.# Edges
COLLAB	5,000	3	74.49	2,457.78
D&D	1,178	2	284.32	715.66
PROTEINS	1,113	2	39.06	72.82
NCI1	4,110	2	29.87	32.30

where a discriminator $\mathcal{D} : \mathbb{R}^f \times \mathbb{R}^f \rightarrow \mathbb{R}$, is employed to represent the probability scores of the local-global pair. The negative samples are drawn by combining the summary vector $\vec{s}$ with the local features $\vec{\tilde{h}_i}$ from other graphs. Through minimizing these log-expectation terms, our model can effectively extract useful local and global information of the input graph based on the mutual information maximization.

3 EXPERIMENTS

We evaluate the graph representation learned from UHGR on both graph classification and node classification tasks. In each case, UHGR is used to learn graph and node representations in a fully unsupervised manner. The graph and node classification tasks are performed by directly feeding the learned representations into simple linear classifiers. We also conduct the visualization experiments on learned representations to verify whether it's reasonable to assign clusters in an unsupervised manner.

3.1 Data sets

To evaluate the ability of UHGR to learn hierarchical representations from arbitrary complex graphs, we perform it on a variety of real-world graphs chosen from the commonly used benchmarks. For the node classification task, we consider the transductive learning setting and choose three standard data sets, Cora, Citeseer, and Pubmed [30], as summarized in Table 1. We employ the same training, validation and testing settings as those in DGI [39], and report the node classification accuracy on the testing data, averaged over 50 runs of training. For graph classification task, we use protein data sets including D&D [6, 32] and PROTEINS [2, 6], the chemical molecules data set NCI1 [32, 41], and the scientific collaboration data set COLLAB [42]. More information on these data sets is shown in Table 2. For this graph classification task, we perform 10-fold cross-validation to evaluate the performance, and apply the average over 10 folds as the final accuracy result. The visualization experiments are conducted on the data sets for graph classification tasks. We feed the original graph to output a coarser one based on the learned hierarchical cluster assignments.

3.2 Experimental setup

As discussed in section 2, UHGR includes encoder module, graph pooling module and discriminator module. The encoder module encodes node representations using one GAT layer or one GCN layer. During the graph pooling module, we apply two DIFFPOOL layers to all of the data sets. Three GCN layers are performed between these two DIFFPOOL layers. In the hierarchical cluster setting, the number of clusters after DIFFPOOL layer is set be to 10-30% of the number of nodes or clusters before pooling. The Readout function in the discriminator module is built on the top of the DIFFPOOL architecture, which enables us to learn the hierarchical graph representations. Finally, the discriminator module relies on the mutual information maximization to achieve the unsupervised graph learning. We also apply Batch normalization [17] after each layer. All models are trained for 1000 epochs with early stopping applied when the validation performance stops improving. We apply PyTorch framework [26] to build graph neural network model and run it on NVIDIA Tesla V100 GPU. In order to demonstrate the effectiveness of our proposed model, we evaluate it on the following three tasks: node classification, graph classification, and analysis of hierarchical cluster assignment.

Reproducibility: Our source code and experimental data are available at https://github.com/ifding/uhgr.

3.3 Results for Node Classification

Table 3 lists the node classification results on data sets Cora, Citeseer and Pubmed using our method and other existing methods. For the operation of node embeddings, we test two different GNN module variants: GATs and GCNs. The GATs module outperforms GCNs on most of the benchmarks, indicating that self-attention mechanism is more suitable for capturing local structural information. For the Cora and Citeseer data sets, we set both hidden dimension and output dimension to 320 and 400, respectively. And for the Pubmed data set, 128-dimensional hidden size and output size for GCN model and 100-dimensional hidden size and output size for GAT model are tested in our experiments. The node representations with larger hidden dimension and output dimension may be more powerful, and will be further optimized in future work. According to the results, our model achieves better classification performance than DeepWalk, and obtains comparable performance with supervised learning methods.

3.4 Results for Graph Classification

Table 4 compares the graph classification performance of our unsupervised learning method with other supervised learning baselines on datasets COLLAB, D&D, PROTEINS and NCI-1. The results show that our unsupervised method obtains similar performance as DIFFPOOL method on the PROTRINS benchmark and achieves comparable results with supervised methods, e.g. GRAPHSAGE, indicating that our method can learn useful graph representations even without graph labels. We also find that GAT-UHGR model performs better than GCN-UHGR model on the datasets COLLAB and PROTEINS, and performs worse than GCN-UHGR model only on the D&D dataset. This suggests that different graph datasets need different Encoder layer to capture useful representations in order to achieve better classification performance. Compared with other

Table 3: Node classification accuracies using different methods on datasets Cora, Citeseer and Pubmed. First column lists the type of data available during each graph representation learning method (X: node features, A: adjacency matrix, Y: node labels, X,A: unsupervised node representation learning, X,A,Y: supervised node classification).

Available data	Method	Cora	Citeseer	Pubmed
X	Raw features	47.9 ± 0.4%	49.3 ± 0.2%	69.1 ± 0.3%
A	DeepWalk	67.2%	43.2%	65.3%
X, A	DeepWalk + features	70.7 ± 0.6%	51.4 ± 0.5%	74.3 ± 0.9%
X, A	DGI	82.3 ± 0.6%	71.8 ± 0.7%	76.8 ± 0.6%
X, A, Y	GCN	81.5%	70.3%	79.0%
X, A, Y	GAT	83.0 ± 0.7%	72.5 ± 0.7%	79.0 ± 0.3%
X, A	GAT-UHGR (ours)	78.5 ± 0.1%	62.6 ± 0.3%	77.4 ± 0.6%
X, A	GCN-UHGR (ours)	76.7 ± 0.1%	62.5 ± 0.1%	75.1 ± 0.3%

Table 4: Graph classification accuracies using different methods on datasets COLLAB, D&D, PROTEINS and NCI-1. First column lists the type of data available during each graph representation learning method (X: node features, A: adjacency matrix, Y: node labels, X,A: unsupervised graph representation learning, X,A,Y: supervised graph classification).

Available data	Method	COLLAB	D&D	PROTEINS	NCI-1
X, A, Y	GRAPHSAGE	68.3%	75.4%	70.5%	-
X, A, Y	SET2SET	71.8%	78.1%	74.3%	-
X, A, Y	DIFFPOOL	75.5%	80.6%	76.3%	79.3%
X, A	graph2vec	-	-	75.4 %	75.0 %
X, A	GAT-UHGR (ours)	67.4%	75.6%	75.9%	66.6%
X, A	GCN-UHGR (ours)	66.9%	77.4%	74.7%	66.6%

unsupervised model, e.g., graph2vec [25], GAT-UHGR model obtains comparable classification results on PROTEIN data set. However, graph2vec utilizes a SVM classifier to perform 1024-dimensional embeddings of graphs, where our method directly uses the graph representations to train and test a simple linear classifier. For the embedding dimensions, we simply set it to 20-360 to demonstrate the validity of the learned hierarchical representations, and doesn't further optimize this hyperparameter to achieve better classification performance due to hardware limitations.

3.5 Visualization of hierarchical representation

In addition to generating useful representation for classification tasks, our model can also create meaningful and interpretable representations in a hierarchical way. To evaluate the meanings of the learned hierarchical graphs, we visualize the cluster assignments after the DIFFPOOL layer. Figure 2 shows the visualization of node assignments on the graphs from different data sets. Different node colors represent different node cluster labels from cluster assignment probabilities. Figure 2 (a) is the node assignment on COLLAB data set and it is clear that our model can capture the hierarchical structure in these graphs. From Figure 2 (b) and (c), we also observe that many meaningful structures, including clique-like, tree-like and cycle-like structures, are captured by the model. This is because the DIFFPOOL layer computes the node assignment based on the node feature matrix and adjacency matrix, thus the input nodes with similar features and local structure obtain similar node assignment. Even if the subgraphs with similar patterns are far away, our model can still assign them into the same cluster. In general, our unsupervised learning method based on mutual information can capture different hierarchical structures.

4 RELATED WORK

Graph Neural Network. A wide variety of graph neural networks have been applied in node classification [10, 20, 38] and graph classification tasks [5, 7, 22, 43, 44] in recent years. In node classification, GAT [38] stacks masked self-attentional layers to classify a node by attending over its neighbors in different weights. LGCN [10] builds a trainable graph convolutional layer to select a fixed number of neighboring nodes in order to transform graph data into grid-like data, which is suitable for typical convolutional operations. PPNP [20] combines graph convolutional networks (GCN) and PageRank to overcome the problem that the size of the observed neighborhood of a node is difficult to extend. In graph classification, the main challenge is to build a useful low-dimensional graph representation based on the node embeddings of the entire graph. One straightforward solution, presented by Duvenaud *et al.* [7] and Veličković *et al.* [39] is to sum or average a graph's node embeddings. However, this solution ignores the structural information of graphs and considers that all nodes contribute the same weight to the calculation of graph representation. Therefore, DiffPool [43] is proposed for graph classification that can learn hierarchical graph representations with a graph pooling module. Although this method solves the problem that existing GNN methods are flat and ignore hierarchical structure of graphs, it needs to learn under the supervision of graph-level labels. In addition, the real-world graphs are

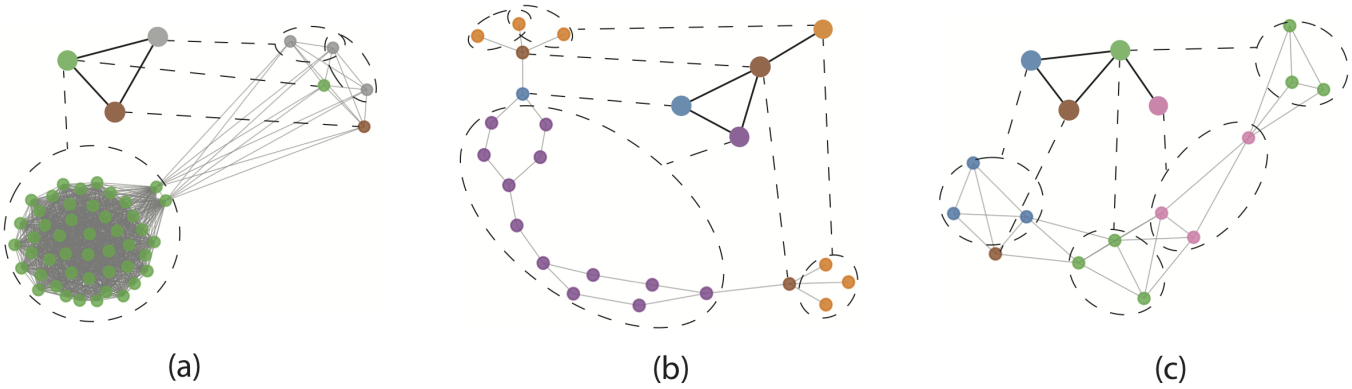

Figure 2: Visualization of hierarchical cluster assignment on data sets (a) COLLAB, (b) NCI1, (c) PROTEINS. The nodes of same color are merged into one cluster in the next layer and the dotted lines represents the cluster membership.

usually large and noisy, GAM [22] is proposed for the attention-based graph classification, which utilizes the attention mechanism to focus on small but informative parts of graphs. Combining local and global information on (hyper)graphs in the hierarchical setting has a long successful history. For example, in computational optimization domain, the multiscale solvers for (hyper)graph partitioning [31], separators [13], and ordering [28] are among the top state of the art methods that preserve excellent time/quality trade-off. However, all of these approaches depend on task-specific information to learn node embeddings or graph embeddings. In addition, most of them ignore the hierarchical representation of graphs, and thus have limited capabilities of capturing the natural structures of the real-world graphs [43].

Graph Representation Learning. Learning a high quality representation not only enables us to capture the latent variables of the data [1], but also helps improve the performance of downstream tasks. For graph-structured data, the learned low-dimensional representations (embeddings) can encode information of a graph's nodes, or the entire graph in the case of the GNN model. Many of the existing graph representations are focused on node embeddings by using random walk based objectives [12, 14, 27]. In addition, LINE [35] and FOBE/HOBE [33] focus on modelling first-order and second-order relationships between node neighborhoods to learn node embeddings and graph embeddings. VERSE [36] is a simple graph embedding framework based on similarity measures. Glimer *et al.* [11] propose a common framework to learn message passing algorithms and aggregate the node embeddings. Janossy Pooling [24] is a permutation-invariant aggregator function to learn node embeddings. Veličković *et al.* [39] propose an alternative unsupervised node embedding method based on mutual information [39]. HARP [3] proposes a hierarchical paradigm to learn low-dimensional representations of a graph's nodes. This paradigm utilizes a smaller graph that approximates the original global structure to obtain good initializations for learning representations of the original graph.

Additional research focuses on learning representations of entire graphs in an unsupervised manner, which is quite different from the task of node embedding. In node embedding, the goal is to learn a low-dimensional vector to represent a node independently of supervised label information (*e.g.*, node labels and graph labels). Graph2vec [25] is an unsupervised graph embedding method inspired by the document embedding models [21], but may not capture global structure, as this method only uses subtrees for graph embeddings. Taheri *et al.* [34] generate sequences from graphs and train a long short-term memory (LSTM) autoencoder model to embed these graph sequences into continuous vectors. The LSTM network cannot be operated in parallel and is not appropriate to model large graphs. Some recent approaches have proposed applying the attention mechanism on graphs [4, 22] that can determine which parts of the graph should have more attention. Yet the attention mechanism only focuses on local information which is not enough to achieve satisfactory node or graph representations. Recently, BayesPool [37] is proposed to use variational Bayes based on an encoder-decoder architecture to learn hierarchical graph representations in an unsupervised manner. Using Encoder-decoder architecture leads to this method being overly focused on node-based details, rather than more high-level node/graph embeddings. Different from previous representation learning methods, in this work we use an unsupervised learning framework based on mutuainformation with contrastive loss, to learn hierarchical graph representations.

5 CONCLUSION

In this paper, we propose an unsupervised hierarchical representation learning model based on mutual information, UHGR, to learn node embeddings and graph embeddings. The mutual information maximization between global representation and local parts of the graphs can encourage the model to learn related structural information in all locations. This unsupervised learning model is able to learn task-independent graph representations. In addition, it can learn hierarchical graph representation, which is meaningful and easy to interpret. To demonstrate the effectiveness of the model, we perform node classification and graph classification tasks based on the learned representations. The results show that our *unsupervised* model can achieve comparable results with the *supervised* methods on several tested data sets. Finally, through visualization of the hierarchical cluster assignment, we show that our model is

able to generate hierarchical representations by clustering different meaningful structures which increases interpretability.

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

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

[3] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. Harp: Hierarchical representation learning for networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[4] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, and Jimeng Sun. 2017. GRAM: graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 787–795.

[5] Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*. 2702–2711.

[6] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.

[7] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*. 2224–2232.

[8] Hanyin Fang, Fei Wu, Zhou Zhao, Xinyu Duan, Yueting Zhuang, and Martin Ester. 2016. Community-based question answering via heterogeneous social network learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[9] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*. 6530–6539.

[10] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1416–1424.

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

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[13] William W Hager, James T Hungerford, and Ilya Safro. 2018. A multilevel bilinear programming algorithm for the vertex separator problem. *Computational Optimization and Applications* 69, 1 (2018), 189–223.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.

[16] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[17] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[18] Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. 2017. Predicting organic reaction outcomes with weisfeiler-lehman network. In *Advances in Neural Information Processing Systems*. 2607–2616.

[19] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[20] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *Seventh International Conference on Learning Representations*.

[21] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.

[22] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1666–1674.

[23] Alessandro Lusci, Gianluca Pollastri, and Pierre Baldi. 2013. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of chemical information and modeling* 53, 7 (2013), 1563–1575.

[24] Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. 2018. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900* (2018).

[25] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[28] Dorit Ron, Ilya Safro, and Achi Brandt. 2011. Relaxation-based coarsening and multiscale graph organization. *SIAM Multiscale Modeling & Simulation* 9, 1 (2011), 407–423.

[29] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. 2017. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*. 991–1001.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[31] Ruslan Shaydulin, Jie Chen, and Ilya Safro. 2019. Relaxation-based coarsening for multilevel hypergraph partitioning. *SIAM Multiscale Modeling & Simulation* 17, 1 (2019), 482–506.

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

[33] Justin Sybrandt and Ilya Safro. 2019. FOBE and HOBE: First-and High-Order Bipartite Embeddings. *arXiv preprint arXiv:1905.10953* (2019).

[34] Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. 2018. Learning graph representations with recurrent neural network autoencoders. In *Proc. KDD Deep Learn. Day*. 1–8.

[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[36] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference*. 539–548.

[37] Shashanka Ubaru and Jie Chen. 2020. Unsupervised Hierarchical Graph Representation Learning with Variational Bayes. https://openreview.net/forum?id=BkgGJlBFPS

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[39] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[40] An Hoa Vu, Nils Ole Tippenhauer, Binbin Chen, David M Nicol, and Zbigniew Kalbarczyk. 2014. CyberSAGE: a tool for automatic security assessment of cyber-physical systems. In *International Conference on Quantitative Evaluation of Systems*. Springer, 384–387.

[41] Nikil Wale, Ian A Watson, and George Karypis. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems* 14, 3 (2008), 347–375.

[42] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1365–1374.

[43] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*. 4805–4815.

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