Unsupervised Inductive Graph-Level Representation Learning via Graph-Graph Proximity

Yunsheng Bai[1] , **Hao Ding**[2] , **Yang Qiao**[1] , **Agustin Marinovic**[1] , **Ken Gu**[1] ,
Ting Chen[1] , **Yizhou Sun**[1] and **Wei Wang**[1]
[1]University of California, Los Angeles
[2]Purdue University
yba@ucla.edu, ding209@purdue.edu, angelinana0408@gmail.com, amarinovic@ucla.edu,
ken.qgu@gmail.com, {tingchen, yzsun, weiwang}@cs.ucla.edu

Abstract

We introduce a novel approach to graph-level representation learning, which is to embed an entire graph into a vector space where the embeddings of two graphs preserve their graph-graph proximity. Our approach, UGRAPHEMB, is a general framework that provides a novel means to performing graph-level embedding in a completely unsupervised and inductive manner. The learned neural network can be considered as a function that receives any graph as input, either seen or unseen in the training set, and transforms it into an embedding. A novel graph-level embedding generation mechanism called Multi-Scale Node Attention (MSNA), is proposed. Experiments on five real graph datasets show that UGRAPHEMB achieves competitive accuracy in the tasks of graph classification, similarity ranking, and graph visualization.

1 Introduction

Recent years we have witnessed the great popularity of graph representation learning with success in not only node-level tasks such as node classification [Kipf and Welling, 2016a] and link prediction [Zhang and Chen, 2018], but also graph-level tasks such as graph classification [Ying *et al.*, 2018] and graph similarity/distance computation [Bai *et al.*, 2019].

There has been a rich body of work [Belkin and Niyogi, 2003; Tang *et al.*, 2015; Qiu *et al.*, 2018] on node-level embeddings that turn each node in a graph into a vector preserving node-node proximity (similarity/distance). Most of these models are unsupervised and demonstrate superb performance in node classification and link prediction. It is natural to raise the question: Can we embed an entire graph into a vector in an unsupervised way, and how? However, most existing methods for graph-level embeddings assume a supervised model [Ying *et al.*, 2018; Zhang and Chen, 2019], with only a few exceptions, such as GRAPH KERNELS [Yanardag and Vishwanathan, 2015] and GRAPH2VEC [Narayanan *et al.*, 2017]. GRAPH KERNELS typically count subgraphs for a given graph and can be slow. GRAPH2VEC is transductive (non-inductive), i.e. it does not naturally generalize to unseen graphs outside the training set.

A key challenge facing designing an unsupervised graph-level embedding model is the lack of graph-level signals in the training stage. *Unlike node-level embedding which has a long history in utilizing the link structure of a graph to embed nodes, there lacks such natural proximity (similarity/distance) information between graphs*. Supervised methods, therefore, typically resort to graph labels as guidance and use aggregation based methods, e.g. average of node embeddings, to generate graph-level embeddings, with the implicit assumption that good node-level embeddings would automatically lead to good graph-level embeddings using only "***intra*-graph** information" such as node attributes, link structure, etc.

However, this assumption is problematic, as simple aggregation of node embeddings can only preserve limited graph-level properties, which is, however, often insufficient in measuring graph-graph proximity ("***inter*-graph**" information). Inspired by the recent progress on graph proximity modeling [Ktena *et al.*, 2017; Bai *et al.*, 2019], we propose a novel framework, **UGRAPHEMB** (*U*nsupervised *Graph*-level *Emb*bedding) that employs multi-scale aggregations of node-level embeddings, guided by the graph-graph proximity defined by well-accepted and domain-agnostic graph proximity metrics such as Graph Edit Distance (GED) [Bunke, 1983], Maximum Common Subgraph (MCS) [Bunke and Shearer, 1998], etc.

The goal of UGRAPHEMB is to learn high-quality ***graph-level*** representations in a completely ***unsupervised*** and ***inductive*** fashion: During training, it learns a function that maps a graph into a universal embedding space best preserving graph-graph proximity, so that after training, any new graph can be mapped to this embedding space by applying the learned function. Inspired by the recent success of pre-training methods in the text domain, such as ELMO [Peters *et al.*, 2018], BERT [Devlin *et al.*, 2018], and GPT [Radford *et al.*, 2018]. we further fine-tune the model via incorporating a supervised loss, to obtain better performance in downstream tasks, including but not limited to:

- **Graph classification**. The embeddings can be fed into any classification model such as logistic regression for graph classification.
- **Graph similarity ranking**. The embeddings learned by UGRAPHEMB preserve the graph-graph proximity by design, and for a graph query, a ranked list of graphs that

are similar to the query can be retrieved.

- **Graph visualization**. The embeddings can be projected into a 2-D space for graph visualization, where each graph is a point. It renders human insights into the dataset and facilitates further database analysis.

In summary, our contributions are three-fold:

1. We formulate the problem of unsupervised inductive graph-level representation learning, and make an initial step towards pre-training methods for graph data. We believe that, given the growing amount of graph datasets of better quality, this work would benefit future works in pre-training methods for graphs.
2. We provide a novel framework, UGRAPHEMB, to generate ***graph-level*** embeddings in a completely ***unsupervised*** and ***inductive*** fashion, well preserving graph proximity. A novel **Multi-Scale Node Attention (MSNA)** mechanism is proposed to generate graph-level embeddings.
3. We conduct extensive experiments on five real network datasets to demonstrate the superb quality of the embeddings by UGRAPHEMB.

2 The Proposed Approach: UGRAPHEMB

We present the overall architecture of our unsupervised inductive graph-level embedding framework UGRAPHEMB in Figure 1. The key novelty of UGRAPHEMB is the use of graph-graph proximity. To preserve the proximity between two graphs, UGRAPHEMB generates one embedding per graph from node embeddings using a novel mechanism called **Multi-Scale Node Attention (MSNA)**, and computes the proximity using the two graph-level embeddings.

2.1 Inductive Graph-Level Embedding

Node Embedding Generation

For each graph, UGRAPHEMB first generates a set of node embeddings. There are two major properties that the node embedding model has to satisfy:

- **Inductivity**. The model should learn a function such that for any new graph unseen in the training set, the learned function can be applied to the graph, yielding its node embeddings.
- **Permutation-invariance**. The same graph can be represented by different adjacency matrices by permuting the order of nodes, and the node embedding model should not be sensitive to such permutation.

Among various node embedding models, neighbor aggregation methods based on Graph Convolutional Networks (GCN) [Kipf and Welling, 2016a] are permutation-invariant and inductive. The reason is that the core operation, graph convolution, updates the representation of a node by aggregating the embedding of itself and the embeddings of its neighbors. Since the aggregation function treats the neighbors of a node as a set, the order does not affect the result.

A series of neighbor aggregation methods have been proposed with different ways to aggregate neighbor information, e.g. GRAPHSAGE [Hamilton *et al.*, 2017], GAT [Velickovic *et al.*, 2018], GIN [Xu *et al.*, 2019], etc. Since UGRAPHEMB is a general framework for *graph*-level embeddings, and all these models satisfy the two properties, any one of these methods can be integrated. We therefore adopt the most recent and state-of-the-art method, Graph Isomorphism Network (GIN) [Xu *et al.*, 2019], in our framework:

$$\boldsymbol{u_i^{(k)}} = \mathrm{MLP}^{(k)}_{\boldsymbol{W_k}}\left(\left(1+\epsilon^{(k)}\right)\cdot \boldsymbol{u_i^{(k-1)}} + \sum_{j\in\mathcal{N}(i)} \boldsymbol{u_j^{(k-1)}}\right) \tag{1}$$

where $\boldsymbol{u_i}$ is the representation of node i, $\mathcal{N}(i)$ is the set of neighbors of node i, $\mathrm{MLP}^{(k)}_{\boldsymbol{W_k}}$ denotes multi-layer perceptrons for the k-th GIN layer with learnable weights $\boldsymbol{W_k}$, and ϵ is a scalar that can either be learned by gradient descent or be a hyperparameter. GIN has been proven to be theoretically the most powerful GNN under the neighbor aggregation framework [Xu *et al.*, 2019].

Graph Embedding Generation

After node embeddings are generated, UGRAPHEMB generates one embedding per graph using these node embeddings. Existing methods are typically based on aggregating node embeddings, by either a simple sum or average, or some more sophisticated way to aggregate.

However, since our goal is to embed each graph as a single point in the embedding space that preserves graph-graph proximity, the graph embedding generation model should:

- **Capture structural difference at multiple scales**. Applying a neighbor aggregation layer on nodes such as GIN cause the information to flow from a node to its direct neighbors, so sequentially stacking K layers would cause the final representation of a node to include information from its K-th order neighbors. However, after many neighbor aggregation layers, the learned embeddings could be too coarse to capture the structural difference in small local regions between two similar graphs. Capturing structural difference at multiple scales is therefore important for UGRAPHEMB to generate high-quality graph-level embeddings.
- **Be adaptive to different graph proximity metrics**. UGRAPHEMB is a general framework that should be able to preserve the graph-graph proximity under any graph proximity metric, such as GED and MCS. A simple aggregation of node embeddings without any learnable parameters limits the expressive power of existing graph-level embedding models.

To tackle both challenges in the graph embedding generation layer, we propose the following **Multi-Scale Node Attention (MSNA)** mechanism. Denote the input node embeddings of graph $\mathcal{G}$ as $\boldsymbol{U_\mathcal{G}} \in \mathbb{R}^{N\times D}$, where the n-th row, $\boldsymbol{u_n} \in \mathbb{R}^D$ is the embedding of node n. The graph level embedding is obtained as follows:

$$\boldsymbol{h_\mathcal{G}} = \mathrm{MLP}_{\boldsymbol{W}}\left(\Big\Vert_{k=1}^{K} \mathrm{ATT}_{\boldsymbol{\Theta}^{(k)}}(\boldsymbol{U_\mathcal{G}})\right) \tag{2}$$

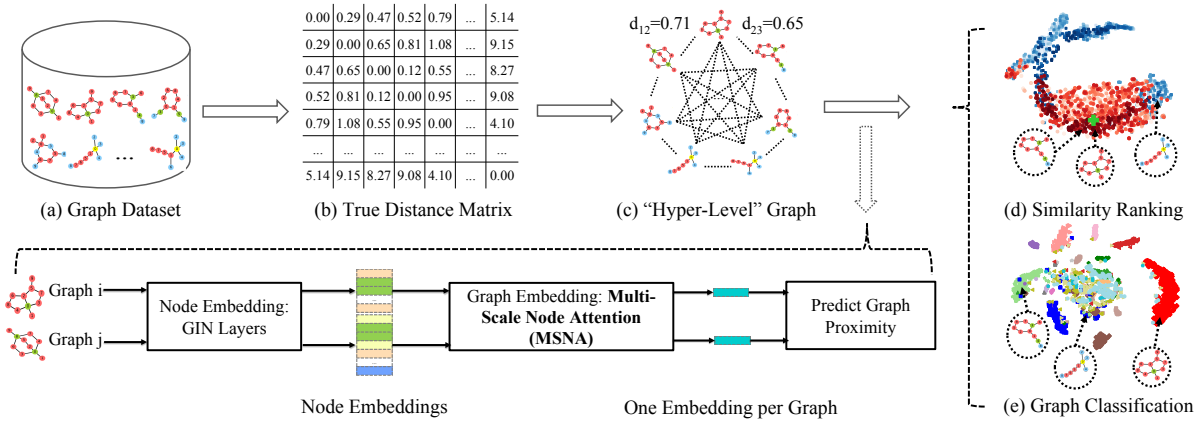

Figure 1: Overview of UGRAPHEMB. (a) Given a set of graphs, (b) UGRAPHEMB first computes the graph-graph proximity scores (normalized distance scores in this example), (c) yielding a "hyper-level graph" where each node is a graph in the dataset, and each edge has a proximity score associated with it, representing its weight/strength. UGRAPHEMB then trains a function that maps each graph into an embedding which preserves the proximity score. The bottom flow illustrates the details of graph-level embedding generation. (d) After embeddings are generated, similarity ranking can be performed. The green "+" sign denotes the embedding of an example query graph. Colors of dots indicate how similar a graph is to the query based on the ground truth (from red to blue, meaning from the most similar to the least similar). (e) Finally, UGRAPHEMB can perform fine-tuning on the proximity-preserving graph-level embeddings, adjusting them for the task of graph classification. Different colors represent different graph labels in the classification task.

where $\|$ denotes concatenation, K denotes the number of neighbor aggregation layers, ATT denotes the following multi-head attention mechanism that transforms node embeddings into a graph-level embedding, and $\text{MLP}_{\boldsymbol{W}}$ denotes multi-layer perceptrons with learnable weights $\boldsymbol{W}$ applied on the concatenated attention results.

The intuition behind Equation 2 is that, instead of only using the node embeddings generated by the last neighbor aggregation layer, we use the node embeddings generated by each of the K neighbor aggregation layers.

ATT is defined as follows:

$$\text{ATT}_{\boldsymbol{\Theta}}(\boldsymbol{U_G}) = \sum_{n=1}^{N} \sigma(\boldsymbol{u_n}^T \text{ReLU}(\boldsymbol{\Theta}(\frac{1}{N}\sum_{m=1}^{N} \boldsymbol{u_m})))\boldsymbol{u_n}. \quad (3)$$

where N is the number of nodes, σ is the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$, and $\boldsymbol{\Theta^{(k)}} \in \mathbb{R}^{D\times D}$ is the weight parameters for the k-th node embedding layer.

The intuition behind Equation 3 is that, during the generation of graph-level embeddings, the attention weight assigned to each node should be adaptive to the graph proximity metric. To achieve that, the weight is determined by both the node embedding $\boldsymbol{u_n}$, and a learnable graph representation. The learnable graph representation is adaptive to a particular graph proximity via the learnable weight matrix $\boldsymbol{\Theta^{(k)}}$.

2.2 Unsupervised Loss via Inter-Graph Proximity Preservation

Definition of Graph Proximity

The key novelty of UGRAPHEMB is the use of graph-graph proximity. It is important to select an appropriate graph proximity (similarity/distance) metric. We identify three categories of candidates:

- **Proximity defined by graph labels**.
  For graphs that come with labels, we may treat graphs of the same label to be similar to each other. However, such proximity metric may be too coarse, unable to distinguish between graphs of the same label.
- **Proximity given by domain knowledge or human experts**.
  For example, in drug-drug interaction detection [Ma *et al.*, 2018], a domain-specific metric to encode compound chemical structure can be used to compute the similarities between chemical graphs. However, such metrics do not generalize to graphs in other domains. Sometimes, this information may be very expensive to obtain. For example, to measure brain network similarities, a domain-specific preprocessing pipeline involving skull striping, band-pass filtering, etc. is needed. The final dataset only contains networks from 871 humans [Ktena *et al.*, 2017].
- **Proximity defined by domain-agnostic and well-accepted metrics**.
  Metrics such as Graph Edit Distance (GED) [Bunke, 1983] and Maximum Common Subgraph (MCS) [Bunke and Shearer, 1998] have been widely adopted in graph database search [Yan *et al.*, 2005; Liang and Zhao, 2017], are well-defined and general to any domain.

In this paper, we use GED as an example metric to demonstrate UGRAPHEMB. GED measures the minimum number of edit operations to transform one graph to the other, where an edit operation on a graph is an insertion or deletion of a node/edge or relabelling of a node. Thus, the GED metric takes both the graph structure and the node labels/attributes into account. The supplementary material contain more details on GED.

Prediction of Graph Proximity

Once the proximity metric is defined, and the graph-level embeddings for $\mathcal{G}_i$ and $\mathcal{G}_j$ are obtained, denoted as $\boldsymbol{h}_{\mathcal{G}_i}$ and $\boldsymbol{h}_{\mathcal{G}_j}$, we can compute the similarity/distance between the two graphs.

Multidimensional scaling (MDS) is a classic form of dimensionality reduction [Williams, 2001]. The idea is to embed data points in a low dimensional space so that their pairwise distances are preserved, e.g. via minimizing the loss function

$$\mathcal{L}(\boldsymbol{h_i}, \boldsymbol{h_j}, d_{ij}) = (||\boldsymbol{h_i} - \boldsymbol{h_j}||_2^2 - d_{ij})^2 \quad (4)$$

where $\boldsymbol{h_i}$ and $\boldsymbol{h_j}$ are the embeddings of points i and j, and d_{ij} is their distance.

Since GED is a well-defined graph distance metric, we can minimize the difference between the predicted distance and the ground-truth distance:

$$\mathcal{L} = \mathbb{E}_{(i,j)\sim\mathcal{D}}(\hat{d_{ij}} - d_{ij})^2 \quad (5)$$

$$= \mathbb{E}_{(i,j)\sim\mathcal{D}}(||\boldsymbol{h_{\mathcal{G}_i}} - \boldsymbol{h_{\mathcal{G}_j}}||_2^2 - d_{ij})^2. \quad (6)$$

where (i, j) is a graph pair sampled from the training set and d_{ij} is the GED between them.

Alternatively, if the metric is similarity, such as in the case of MCS, we can use the following loss function:

$$\mathcal{L} = \mathbb{E}_{(i,j)\sim\mathcal{D}}(\hat{s_{ij}} - s_{ij})^2 \quad (7)$$

$$= \mathbb{E}_{(i,j)\sim\mathcal{D}}(\boldsymbol{h}_{\mathcal{G}_i}^T \boldsymbol{h_{\mathcal{G}_j}} - s_{ij})^2. \quad (8)$$

After training, the learned neural network can be applied to any graph, and the graph-level embeddings can facilitate a series of downstream tasks, and can be fine-tuned for specific tasks. For example, for graph classification, a supervised loss function can be used to further enhance the performance.

3 Experiments

We evaluate our model, UGRAPHEMB, against a number of state-of-the-art approaches designed for unsupervised node and graph embeddings, to answer the following questions:

- **Q1** How superb are the graph-level embeddings generated by UGRAPHEMB, when evaluated with downstream tasks including graph classification and similarity ranking?
- **Q2** Do the graph-level embeddings generated by UGRAPHEMB provide meaningful visualization for the graphs in a graph database?
- **Q3** Is the quality of the embeddings generated by UGRAPHEMB sensitive to choices of hyperparamters?

Datasets We evaluate the methods on five real graph datasets, PTC, IMDBM, WEB, NCI109, and REDDIT12K. The largest dataset, REDDIT12K, has 11929 graphs.

3.1 Task 1: Graph Classification

Intuitively, the higher the quality of the embeddings, the better the classification accuracy. Thus, we feed the graph-level embeddings generated by UGRAPHEMB and the baselines into a logistic regression classifier to evaluate the quality: (1) GRAPH KERNELS (GRAPHLET (GK), DEEP GRAPHLET (DGK), SHORTEST PATH (SP), DEEP SHORTEST PATH (DSP), WEISFEILER-LEHMAN (WL), and DEEP WEISFEILER-LEHMAN (DWL)) ; (2) GRAPH2VEC [Narayanan *et al.*, 2017]; (3) NETMF [Qiu *et al.*, 2018]; (4) GRAPHSAGE [Hamilton *et al.*, 2017].

For GRAPH KERNELS, we also try using the kernel matrix and SVM classifier as it is the standard procedure outlined in [Yanardag and Vishwanathan, 2015], and report the better accuracy of the two. For (3) and (4), we try different averaging schemes on node embeddings to obtain the graph-level embeddings and report their best accuracy.

Method	PTC	IMDBM	WEB	NCI109	REDDIT12K
GK	57.26	43.89	21.37	62.06	31.82
DGK	57.32	44.55	23.65	62.69	32.22
SP	58.24	37.01	18.16	73.00	–
DSP	60.08	39.67	22.65	73.26	–
WL	66.97	49.33	26.44	**80.22**	39.03
DWL	70.17	49.95	34.56	**80.32**	39.21
GRAPH2VEC	60.17	47.33	**40.91**	74.26	35.24
NETMF	56.65	30.67	19.71	51.84	23.24
GRAPHSAGE	52.17	34.67	20.38	65.09	25.01
UGRAPHEMB	**72.54**	**50.06**	37.36	69.17	**39.97**
UGRAPHEMB-F	**73.56**	**50.97**	**45.03**	74.48	**41.84**

Table 1: Graph classification accuracy in percent. "-" indicates that the computation did not finish after 72 hours. We highlight the top 2 accuracy in bold.

As shown in Table 1, UGRAPHEMB without fine-tuning, i.e. using only the unsupervised "inter-graph" information, can already achieve top 2 on 3 out of 5 datasets and demonstrates competitive accuracy on the other datasets. With fine-tuning (UGRAPHEMB-F), our model can achieve the best result on 4 out of 5 datasets. Methods specifically designed for graph-level embeddings (GRAPH KERNELS, GRAPH2VEC, and UGRAPHEMB) consistently outperform methods designed for node-level embeddings (NETMF and GRAPHSAGE), suggesting that *good node-level embeddings do* ***not*** *naturally imply good graph-level representations*.

3.2 Task 2: Similarity Ranking

For each dataset, we split it into training, validation, and testing sets by 6:2:2, and report the averaged *Mean Squared Error (mse)*, *Kendall's Rank Correlation Coefficient (τ)* [Kendall, 1938], and *Precision at 10 (p@10)* to test the ranking performance.

Table 2 shows that UGRAPHEMB achieves state-of-the-art ranking performance under all settings except one. This should not be a surprise, because only UGRAPHEMB utilizes the ground-truth GED results collectively determined by BEAM [Neuhaus *et al.*, 2006], HUNGARIAN [Riesen and Bunke, 2009], and VJ [Fankhauser *et al.*, 2011]. UGRAPHEMB even outperforms HED [Fischer *et al.*, 2015], a state-of-the-art approximate GED computation algorithm, under most settings, further confirming its strong ability to generate proximity-preserving graph embeddings by learning from a specific graph proximity metric, which is GED in this case.

3.3 Task 3: Embedding Visualization

Visualizing the embeddings on a two-dimensional space is a popular way to evaluate node embedding methods [Tang *et al.*, 2015]. However, we are among the first to investigate the question: Are the graph-level embeddings generated by a model like UGRAPHEMB provide meaningful visualization?

We feed the graph emebddings learned by all the methods into the visualization tool t-SNE [Maaten and Hinton, 2008]. The three deep graph kernels, i.e. DGK, DSP, and WDL,

Method	PTC		IMDBM		WEB		NCI109		REDDIT12K	
	τ	p@10	τ	p@10	τ	p@10	τ	p@10	τ	p@10
GK	0.291	0.135	0.329	0.421	0.147	0.101	0.445	0.012	0.007	0.009
DGK	0.222	0.103	0.304	0.410	0.126	0.009	0.441	0.010	0.011	0.012
SP	0.335	0.129	0.009	0.011	0.008	0.065	0.238	0.012	–	–
DSP	0.344	0.130	0.007	0.010	0.011	0.072	0.256	0.019	–	–
WL	0.129	0.074	0.034	0.038	0.014	0.246	0.042	0.006	0.089	0.017
DWL	0.131	0.072	0.039	0.041	0.017	0.262	0.049	0.009	0.095	0.023
GRAPH2VEC	0.128	0.188	0.697	0.624	0.014	0.068	0.033	0.127	0.008	0.017
NETMF	0.004	0.012	0.003	0.143	0.002	0.010	0.001	0.008	0.009	0.042
GRAPHSAGE	0.011	0.033	0.042	0.059	0.009	0.010	0.018	0.040	0.089	0.017
BEAM	0.992*	0.983*	0.892*	0.968*	0.963*	0.957*	0.615*	0.997*	0.657*	1.000*
HUNGARIAN	0.755*	0.465*	0.872*	0.825*	0.706*	0.160*	0.667*	0.164*	0.512*	0.808*
VJ	0.749*	0.403*	0.874*	0.815*	0.704*	0.151*	0.673*	0.097*	0.502*	0.867*
HED	0.788	0.433	0.627	0.801	**0.667**	0.291	0.199	0.174	0.199	0.237
UGRAPHEMB	**0.840**	**0.457**	**0.853**	**0.816**	0.618	**0.303**	**0.476**	**0.189**	**0.572**	**0.365**

Table 2: Similarity ranking performance. BEAM, HUNGARIAN, and VJ are three approximate GED computation algorithms returning upper bounds of exact GEDs. We take the minimum GED computed by the three as ground-truth GEDs for training and evaluating all the methods on both Task 1 and 2. Their results are labeled with "$*$". HED is another GED solver yielding lower bounds. "-" indicates that the computation did not finish after 72 hours.

generate the same embeddings as the non-deep versions, but use additional techniques [Yanardag and Vishwanathan, 2015] to modify the similarity kernel matrices, resulting in different classification and ranking performance.

From Figure 2, we can see that UGRAPHEMB can separate the graphs in IMDBM into multiple clusters, where graphs in each cluster share some common substructures.

Such clustering effect is likely due to our use of graph-graph proximity scores, and is thus not observed in NETMF or GRAPHSAGE. For GRAPH KERNELS and GRAPH2VEC though, there are indeed clustering effects, but by examining the actual graphs, we can see that graph-graph proximity is not well-preserved by their clusters (e.g. for WL graph 1, 2 and 9 should be close to each other; for GRAPH2VEC, graph 1, 2, and 12 should be close to each other), explaining their worse similarity ranking performance in Table 2 compared to UGRAPHEMB.

3.4 Parameter Sensitivity of UGRAPHEMB

We evaluate how the dimension of the graph-level embeddings and the percentage of graph pairs with ground-truth GEDs used to train the model can affect the results. We report the graph classification accuracy on IMDBM.

As can be seen in Figure 3, the performance becomes marginally better if larger dimensions are used. For the percentage of training pairs with ground-truth GEDs, the performance drops as less pairs are used. Note that the x-axis is in log-scale. When we only use 0.001% of all the training graph pairs (only 8 pairs with ground-truth GEDs), the performance is still better than many baseline methods, exhibiting impressive insensitivity to data sparsity.

4 Related Work

Unsupervised graph representation learning has a long history. Classic works including NETMF [Qiu *et al.*, 2018], LINE [Tang *et al.*, 2015], DeepWalk [Perozzi *et al.*, 2014], etc., which typically generate an embedding for each node in ***one*** graph. Theoretical analysis shows that many of these works cannot handle embeddings for multiple graphs in the sense that the node embeddings in one graph are not comparable to those in another graph in any straightforward way [Heimann and Koutra, 2017]. A simple permutation of node indices could cause the node embedding to be very different.

More recently, some of the methods based on Graph Convolutional Networks (GCN) [Defferrard *et al.*, 2016; Kipf and Welling, 2016a], such as VGAE [Kipf and Welling, 2016b], satisfy the desired permutation-invariance property. Categorized as "graph autoencoders" [Wu *et al.*, 2019], they also belong to the family of graph neural network methods. Although satisfying the permutation-invariance requirement, these autoencoders are still designed to generate unsuperised node embeddings.

Methods designed for unsupervised graph-level embeddings include GRAPH2VEC [Narayanan *et al.*, 2017] and GRAPH KERNELS [Yanardag and Vishwanathan, 2015], which however are either not based on learning or not inductive. Unlike node-level information which is reflected in the neighborhood of a node, graph-level information is much more limited. A large amount of graph neural network models resort to graph labels as a source of such information, making the models supervised aiming to improve graph classification accuracy specifically, such as DIFFPOOL [Ying *et al.*, 2018], CAPS-GNN [Zhang and Chen, 2019], etc., while UGRAPHEMB learns a function that maps each graph into an embedding that can be used to facilitate many downstream tasks.

5 Conclusion

We present UGRAPHEMB, an end-to-end neural network based framework aiming to embed an entire graph into an embedding preserving the proximity between graphs in the dataset under a graph proximity metric, such as Graph Edit Distance (GED). A novel mechanism for generating graph-

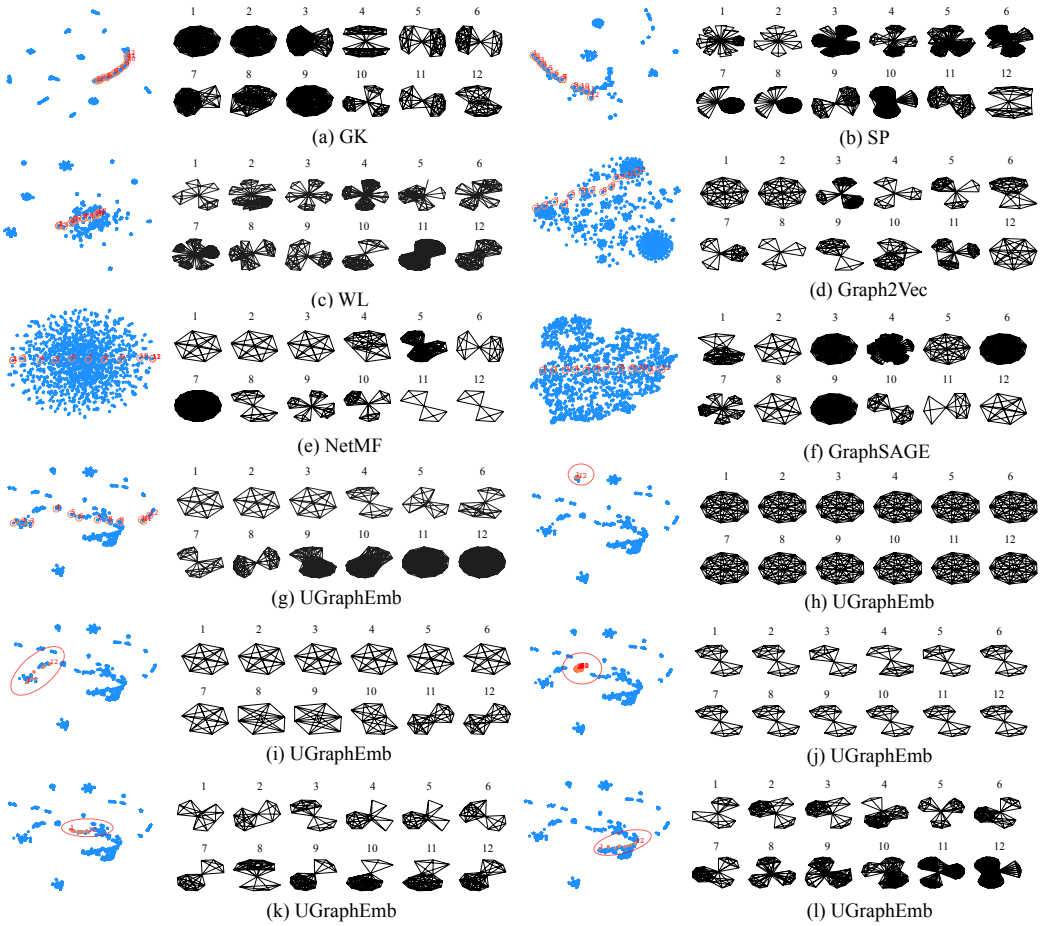

Figure 2: Visualization of the IMDBM dataset. From (a) to (g), for each method, 12 graphs are plotted. For (h) to (l), we focus on UGRAPHEMB: 5 clusters are highlighted in red circles. 12 graphs are sampled from each cluster and plotted to the right.

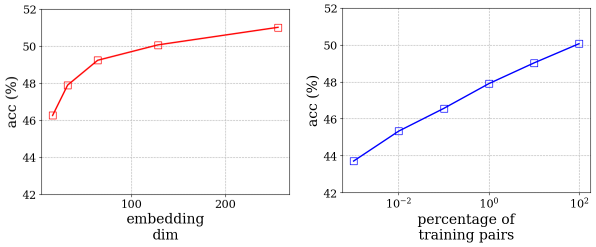

Figure 3: Classification accuracy on the IMDBM dataset w.r.t. the dimension of graph-level embeddings and the percentage of graph pairs used for training.

level embeddings is proposed. Experiments show that the produced graph-level embeddings achieve competitive performance on three downstream tasks: graph classification, similarity ranking, and graph visualization.

Acknowledgements

This work is partially supported by NIH R01GM115833 and U01HG008488, NSF DBI-1565137, DGE-1829071, NSF III-1705169, NSF CAREER Award 1741634, and Amazon Research Award.

References

[Bai *et al.*, 2019] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. *WSDM*, 2019.

[Belkin and Niyogi, 2003] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[Bunke and Shearer, 1998] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3-4):255–259, 1998.

[Bunke, 1983] H Bunke. What is the distance between graphs. *Bulletin of the EATCS*, 20:35–39, 1983.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of

deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Fankhauser *et al.*, 2011] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. Speeding up graph edit distance computation through fast bipartite matching. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 102–111. Springer, 2011.

[Fischer *et al.*, 2015] Andreas Fischer, Ching Y Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. Approximation of graph edit distance based on hausdorff matching. *Pattern Recognition*, 48(2):331–343, 2015.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[Heimann and Koutra, 2017] Mark Heimann and Danai Koutra. On generalizing neural node embedding methods to multi-network problems. In *KDD MLG Workshop*, 2017.

[Kendall, 1938] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[Kipf and Welling, 2016a] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2016.

[Kipf and Welling, 2016b] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[Ktena *et al.*, 2017] Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 469–477. Springer, 2017.

[Liang and Zhao, 2017] Yongjiang Liang and Peixiang Zhao. Similarity search in graph databases: A multi-layered indexing approach. In *ICDE*, pages 783–794. IEEE, 2017.

[Ma *et al.*, 2018] Tengfei Ma, Cao Xiao, Jiayu Zhou, and Fei Wang. Drug similarity integration through attentive multiview graph auto-encoders. *IJCAI*, 2018.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[Narayanan *et al.*, 2017] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *KDD MLG Workshop*, 2017.

[Neuhaus *et al.*, 2006] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 163–172. Springer, 2006.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

[Peters *et al.*, 2018] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *NAACL*, 2018.

[Qiu *et al.*, 2018] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifyingdeepwalk, line, pte, and node2vec. *WSDM*, 2018.

[Radford *et al.*, 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[Riesen and Bunke, 2009] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7):950–959, 2009.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[Williams, 2001] Christopher KI Williams. On a connection between kernel pca and metric multidimensional scaling. In *Advances in neural information processing systems*, pages 675–681, 2001.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ICLR*, 2019.

[Yan *et al.*, 2005] Xifeng Yan, Philip S Yu, and Jiawei Han. Substructure similarity search in graph databases. In *SIGMOD*, pages 766–777. ACM, 2005.

[Yanardag and Vishwanathan, 2015] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *SIGKDD*, pages 1365–1374. ACM, 2015.

[Ying *et al.*, 2018] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *NeurIPS*, 2018.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS*, pages 5171–5181, 2018.

[Zhang and Chen, 2019] Xinyi Zhang and Lihui Chen. Capsule graph neural network. *ICLR*, 2019.