CGMN: A Contrastive Graph Matching Network for Self-Supervised Graph Similarity Learning

Di Jin[1], **Luzhi Wang**[1], **Yizhen Zheng**[2], **Xiang Li**[3], **Fei Jiang**[3], **Wei Lin**[3] and **Shirui Pan**[2*]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Department of Data Science and AI, Faculty of IT, Monash University, Australia
[3]Meituan, Beijing, China
{jindi, wangluzhi}@tju.edu.cn, {yizhen.zheng1, shirui.pan}@monash.edu

Abstract

Graph similarity learning refers to calculating the similarity score between two graphs, which is required in many realistic applications, such as visual tracking, graph classification, and collaborative filtering. As most of the existing graph neural networks yield effective graph representations of a single graph, little effort has been made for jointly learning two graph representations and calculating their similarity score. In addition, existing unsupervised graph similarity learning methods are mainly clustering-based, which ignores the valuable information embodied in graph pairs. To this end, we propose a contrastive graph matching network (CGMN) for self-supervised graph similarity learning in order to calculate the similarity between any two input graph objects. Specifically, we generate two augmented views for each graph in a pair respectively. Then, we employ two strategies, namely cross-view interaction and cross-graph interaction, for effective node representation learning. The former is resorted to strengthen the consistency of node representations in two views. The latter is utilized to identify node differences between different graphs. Finally, we transform node representations into graph-level representations via pooling operations for graph similarity computation. We have evaluated CGMN on eight real-world datasets, and the experiment results show that the proposed new approach is superior to the state-of-the-art methods in graph similarity learning downstream tasks.

1 Introduction

Graph-structured data is a natural representation in coding structures that exist in a variety of domains, including computer vision [Wan *et al.*, 2021], social network [Jin *et al.*, 2021a], NLP [Yu *et al.*, 2021], recommender systems [Jia *et al.*, 2021; Jin *et al.*, 2021b], etc. Graph similarity learning is one of the most important research problems in exploring graph-structured data. It aims to learn a similarity score for a pair of input graphs. Graph similarity learning has a wide range of applications in various scenarios, such as binary code analysis [Xu *et al.*, 2017], collaborative filtering algorithms for recommender systems [Su *et al.*, 2021], etc. Different task scenarios have different similarity metrics, e.g., graph edit distance (GED) [Sanfeliu and Fu, 1983], and maximum common subgraph (MCS) [Bunke, 1997]. The GED problem refers to solving the minimum number of editing operations required to convert one graph to the other, and MCS aims to find the largest induced subgraph common to two given graphs. However, GED and MCS problems are a class of NP-complete problems [Wang *et al.*, 2021; Bai *et al.*, 2021], indicating that solving such problems is computationally expensive. Some traditional methods, such as A* [Neuhaus *et al.*, 2006] and Hungarian [Riesen and Bunke, 2009], can accurately solve the problem of GED, but they have very high computational complexity and thus are hard to be applied on large-scale real datasets. In addition, traditional search algorithms mainly consider the graph structure, while ignoring the rich attribute information contained in the graph, including node features, edge features, etc, limiting their applicability.

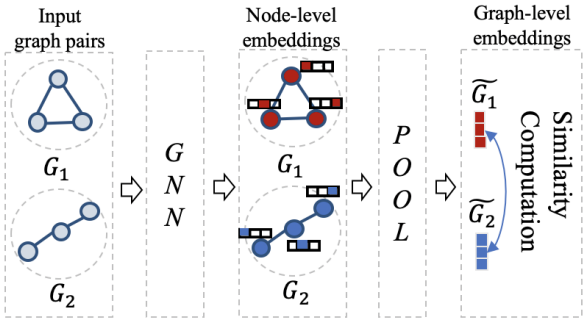


Figure 1: A view of the supervised graph similarity learning model.

Given the great difficulty of computing the graph distances, people begin to use graph neural networks (GNNs) to learn the similarity between two graphs [Li *et al.*, 2019]. But most of the current GNN-based graph similarity learning approaches are supervised and normally adopt a training scheme, which can be summarized as follows. First, the representation of nodes in each graph are learned by using a

*Corresponding author

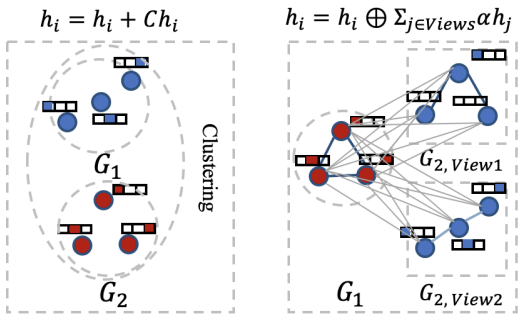


Figure 2: Comparison of node information update process. Left: Clustering-based unsupervised graph similarity learning. Right: Our proposed model CGMN. h_i denotes the embedding of node i, C represents the clustering weight, $\oplus$ is concatenation, and α represents the cross-graph interaction weight.

GNN encoder. Then, graph-level representations are obtained by using a pooling operation. After that, with a distance metric (e.g., cosine similarity and Euclidean distance), the similarity score can be obtained given a pair of graph-level representations. Finally, with the guidance of ground truth, they train the model with the computed loss, which is the difference between the similarity score and the ground truth labels. Fig. 1 shows the details of this process. However, the heavy reliance of these methods on labels hinders them from being applied in real-world applications where the labeling information is usually scarce and expensive to obtain. To address this problem, we opt to design an unsupervised deep graph similarity learning algorithm.

However, the current unsupervised GNN methods mainly focus on node (graph) classification or clustering tasks, and limited works have explored graph similarity learning. There are a few unsupervised graph similarity learning studies, such as GA-MGMC [Wang *et al.*, 2020] and ScGSLC [Li *et al.*, 2021]. The main idea of these works can be summarized as follows. First, these works cluster all graphs and compute the clustering weight C for any two graphs, where C is learned based on the clustering results (i.e., graphs in the same cluster have higher C, and vice versa). Then, node embeddings are updated with C and pooled to generate the graph-level embeddings. And finally, these works use the pseudo labels and the learned similarity score to calculate the loss. Fig. 2 (left) briefly describes the process of updating node information. These clustering-based methods suffer from limitations including the negligence of rich information in cross-graph interaction, as well as the manual choice of the number of clusters. Specifically, they rarely consider the node-to-node matching across graphs, which can provide abundant supervision signals for model training [Li *et al.*, 2019]. Furthermore, they require to manually choose an optimal number of clusters for training, which is essential in clustering but rather difficult to set in practice. Thus, these methods typically lead to suboptimal solutions.

In order to solve the aforementioned problems, we design a novel scheme for unsupervised graph similarity learning, named as **C**ontrastive **G**raph **M**atching **N**etwork (CGMN). We construct a node embedding method based on augmented views and aim to compute graph similarity scores between graphs. We first augment two different views for each graph in a graph pair to provide different contexts. With a designed cross-view interaction module, for a target graph, we weightily aggregate all node embeddings in the other view to obtain the view-level embedding. After that, we conduct cross-graph matching, which outputs a graph-level embedding for the target node, to distill the rich information embodied in cross-graph interaction. Both the view- and graph-level embedding are concatenated to the node embedding, which is used in building contrastiveness between two views to learn node embeddings of the graph.

We expect cross-graph matching of nodes to occur during the learning process, so we interact node information between one graph and views of the other graph, instead of interacting at the end of node representation training. Interacting nodes with different contexts shows the node-to-node similarity from different views. After the node embedding training, we obtain the representation of the whole graph by the pooling module. Similarity scores are calculated for different tasks using graph-level embedding. Fig. 2 (right) illustrates the process of node information update.

Our contributions can be summarized as follows:

- We introduce contrastive learning to the unsupervised graph similarity model, which improves the accuracy of graph similarity through node-to-node matching instead of generally using clustering to simulate similarity calculation.
- We propose *cross-view interactions* and *cross-graph interactions* in unsupervised graph similarity learning in the same framework, which enhances agreement of nodes and fills the gap that two graphs cannot interact with each other in clustering-based methods.
- We conduct experiments on eight real-world datasets, and the experimental results validate the superiority of our proposed new method compared with SOTA.

2 Related Work

2.1 Graph Similarity Learning

Inspired by recent advances in deep learning, computing graph similarity with deep networks has received increasing attention. The first category is supervised graph similarity learning, which is a line of work that uses deep feature encoders to learn the similarity of the input pair of graphs. The learned similarity scores and ground truth are fed into the loss function to optimize the encoder with resulting in an end-to-end learning scheme. For example, GMN [Li *et al.*, 2019], SimGNN [Bai *et al.*, 2019], and H2MN [Zhang *et al.*, 2021] all belong to this category, which use GNNs as their encoder. The second category is the unsupervised graph similarity learning, such as GA-MGMC [Wang *et al.*, 2020], ScGSLC [Li *et al.*, 2021], etc. The process of these works stands to encode the nodes using a deep encoder and update the features of the nodes by using the clustering weights. But they are limited to clustering and thus difficult to learn the cross-graph interaction information of two graphs.

2.2 Graph Contrastive Learning

Contrastive learning is popular in self-supervised graph representation learning, which aims to learn discriminative representations by comparing positive and negative samples [He *et al.*, 2020; Jin *et al.*, 2021c]. For example, Deep Graph Infomax (DGI) [Velickovic *et al.*, 2019] extends Deep Infomax [Hjelm *et al.*, 2019] by comparing nodes and graphs to learn node representations. MVGRL [Hassani and Khasahmadi, 2020] introduces graph diffusion to graph augmentation by comparing different graph structures to learn node representations. GRACE [Zhu *et al.*, 2020] maximizes the agreement of node representations between two augmented graph views. Contrastive learning has shown impressive results in unsupervised graph representation learning, but it is typically used for classification and clustering tasks. There are still no models designed for graph similarity learning tasks as far as we know.

3 Method

3.1 Problem Definition

An undirected attributed graph $G = (V, E, A, X)$ consists of a set of nodes V with $|V| = n$, a set of edges E with $|E| = m$, the adjacency matrix $A \in \mathbb{R}^{n\times n}$ and node attribute matrix $X \in \mathbb{R}^{n\times d}$ with feature dimension d. If there is an edge between node i and node j, $A_{i,j} = 1$, else $A_{i,j} = 0$. For each node, its features are represented as a vector $x \in \mathbb{R}^d$, where x_i denotes feature values of node i. Therefore, $X \in \mathbb{R}^{n\times d}$ denotes the node attribute matrix of the graph and each node has d features. Given two graphs G_1 and G_2 as input, the graph similarity learning is to calculate a similarity score y in order to measure the difference between two graphs in an input graph pair. Different similarity metrics can be defined according to different downstream tasks. Our method tries to learn an encoder to generate graph-level embeddings of two graphs and calculate their similarity score.

3.2 Method Overview

Fig. 3 shows our overall framework CGMN. Firstly, in part (a), two graphs G_1 and G_2 are fed into CGMN. Then, CGMN generates two correlated graph views via graph augmentation techniques for G_1 and G_2 respectively. After that, the original input graphs and their views are embedded into a low-dimensional vector space by a GNN encoder. Secondly, a cross-view interaction strategy (b) and a cross-graph interaction strategy (c) are provided to learn effective node embeddings. In part (d), graph-level representations are obtained by pooling the node embeddings. Finally, CGMN computes similarity using two graph-level representations for the subsequent prediction tasks (e). Details will be introduced in the rest of this section.

Node embedding layer. The siamese network consists of two identical sub-networks and excels at comparing the subtle differences between the two inputs. We integrate a siamese network architecture with contrastive learning-based GNN as the basic node embedding layer of CGMN. Graph augmentation, a key component of graph contrastive learning, is used to generate different views to provide diverse contexts for nodes, which facilitates the optimization of contrastive objectives. We generate two correlated graph views by randomly performing corruption techniques, such as masking node features and removing edges. Positive and negative samples are generated to learn node representations discriminatively by comparing them. After graph augmentation, we consider using a multi-layer GCN to generate node embeddings of the input graphs and their views:

$$H^l = \sigma(\widetilde{A}H^{l-1}W^{l-1}), \tag{1}$$

where H is the set of node embeddings of a graph (view), l is the number of layers, σ is the activation function, $\widetilde{A}$ is the normalized Laplacian matrix, and W is the layer-specific trainable weighted matrix.

Cross-view interaction. Inspired by works on graph contrastive learning, the agreement of corresponding nodes in the two views should be maximized. Specifically, if node $u \in G_{1,View1}$ corresponds to node $v \in G_{1,View2}$, node u's embedding h_u and node v's embedding h_v can be regarded as a pair of positive samples, and embeddings of other nodes in the two views can be naturally regarded as negative samples of h_u [Zhu *et al.*, 2020]. To maximize agreement between node embeddings, we need to expand the similarity between positive samples and reduce the similarity between negative samples. The similarity of two nodes' embeddings in vector space can be defined as:

$$\mathtt{sim}(h_u, h_v) := \mathtt{exp}(\mathtt{cos}\left(h_u, h_v\right)/\tau), \tag{2}$$

here, $\mathtt{cos}(\cdot)$ donates the cosine similarity function, $\mathtt{exp}(\cdot)$ is the exponential function, and τ is a temperature parameter to help the model learn from hard negatives which are closest in distance, but farther than positive examples [Chen *et al.*, 2020].

However, the aforementioned processes mainly consider the corresponding nodes' agreement between views. The similarity between the nodes in the two views has not been fully considered. To address this drawback, we propose a cross-view interaction method that measures node-to-node similarity with cosine similarity, which enriches the self-supervised signals.

$$\hat{h_u} = h_u \oplus \sum_{v\in G_{1,View2}} \mathtt{cos}(h_u, h_v)h_v, \tag{3}$$

where $\oplus$ is the concatenation. Fig. 4 shows the difference the contrastive learning and cross-view interaction.

Cross-view interactions help maintain the monotonic increasing correlation between node similarity and node agreement. If nodes in the two views are corresponding, they have higher cross-view interaction scores than others, which means that their embeddings are close in the vector space and promote the maximization of node agreements.

Cross-graph interaction. Learning node-to-node matching between two graphs is a *key* part of graph similarity learning. If we perform cross-graph matching after the node representation training is completed, we may not be able to better learn the interaction information of the two graphs. In the above process, we augment each graph with two views, which can replace the graph to participate in the cross-graph information interaction during the training process. Specifically, we interact the original graph G_1 with each view of G_2.

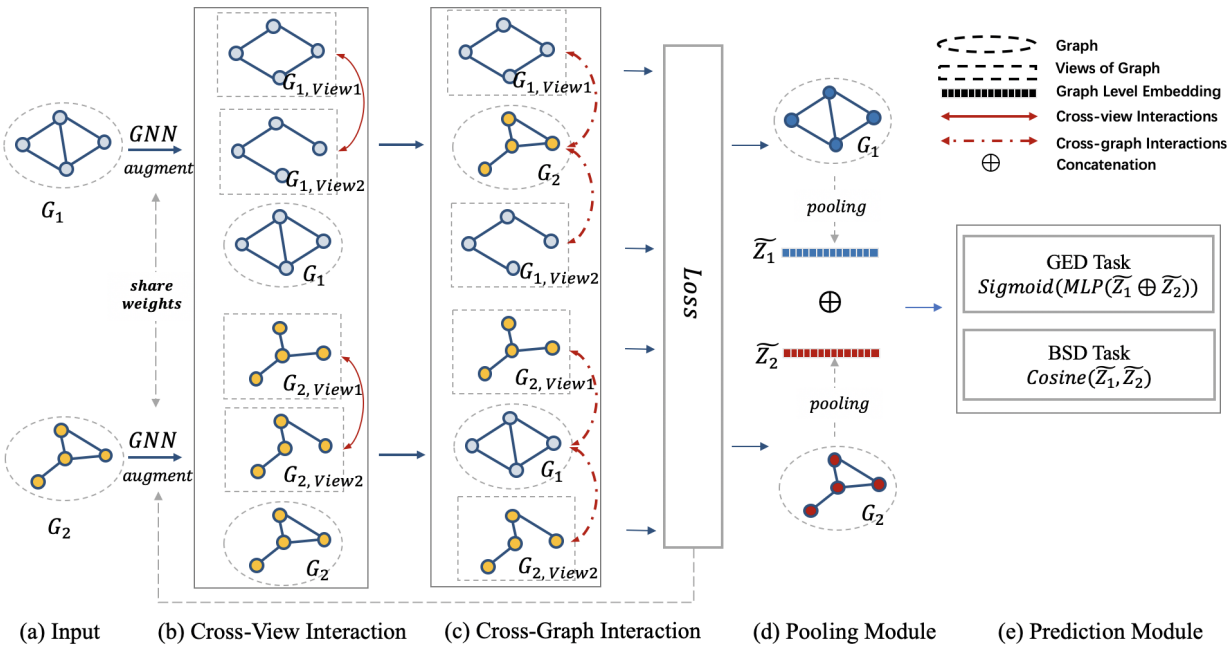


Figure 3: Overview of CGMN. First, we provide a framework to learn the embedding of each node. Second, we propose a cross-graph interaction strategy to match nodes in graph pairs. Third, we aggregate node embeddings to obtain the graph-level representations. Finally, we predict the similarity of graphs for different tasks.

We use the attention mechanism to implement cross-graph information interaction. We design the attention function for computing the similarity score between two node embedding vectors. We use cosine similarity as a concrete form of attention mechanism. Assuming that node u belongs to the graph G_1, then node u's representation with cross-graph information can be calculated as:

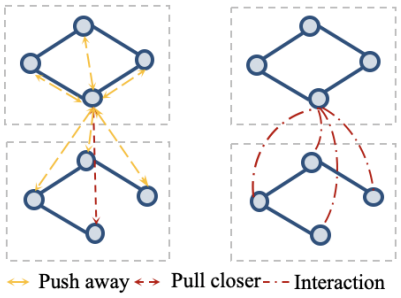
$$h_u^* = \hat{h_u} \oplus \sum_{i \in G_{2,View1}} \cos(\hat{h_u}, \hat{h_i}) \oplus \sum_{j \in G_{2,View2}} \cos(\hat{h_u}, \hat{h_j}). \tag{4}$$

Cross-graph interaction and cross-view interaction are different. The main difference lies in not only the object of action, but also the purpose. The purpose of cross-view interaction is to maximize node agreement in different views, whereas cross-graph interaction aims to learn the node matching between different graphs. After cross-view interaction and cross-graph interaction learning, we construct the self-supervised loss by maximizing the agreement of positive samples and minimizing the agreement of negative samples:

$$loss(h_u^*, h_v^*) = -\log \frac{\mathtt{sim}(h_u^*, h_v^*)}{\mathtt{sim}(h_u^*, h_v^*) + \sum_{k=1}^{N} \mathtt{sim}(h_u^*, h_k^*)}, \tag{5}$$

where u and v are a pair of positive samples and N is the number of negative samples. Both node u and node v need to be calculated, then the total loss can be defined as their average result:

$$\mathcal{L} = \frac{1}{2}\left[loss(h_u^*, h_v^*) + loss(h_v^*, h_u^*)\right]. \tag{6}$$



(a) Contrastive learning. (b) Cross-view interaction.

Figure 4: Difference between (a) contrastive learning and (b) cross-view interaction.

Prediction layer. We calculate graph similarity by graph-level embeddings. For simplicity, we employ mean pooling as the default aggregation function:

$$\tilde{Z} = \mathtt{AVG}(h_u : u \in G), \tag{7}$$

where $\mathtt{AVG}(\cdot)$ denotes the average function.

For different prediction tasks, we can define different methods for calculating similarity scores. In **GED** task, given two graphs G_1 and G_2 as input, $GED(G_1, G_2)$ records the number of edits that transform G_1 to G_2. An edit can be defined as an insertion or a deletion operation of a node or edge. The number of GED is converted to consecutive values in $(0, 1]$ by $exp(-\frac{GED(G_1,G_2)}{|G_1|+|G_2|})$. In our implementation, we use a multi-layer perceptron (MLP) to obtain the GED score, which takes the concatenation of the two graph-level embeddings as input.

The similarity score is then calculated as:

$$y = \texttt{sigmoid}(\texttt{MLP}(\tilde{Z}_1 \oplus \tilde{Z}_2)). \tag{8}$$

The **binary similarity detection (BSD)** task aims to predict whether the given graphs are similar or not. The input two graphs are assigned a label $\hat{y} = \{1, -1\}$ where 1 indicates the input two graphs are similar, and -1 indicates these two graphs are not similar. Our goal is to learn the similarity between the input two graphs, and the calculated similarity score will be classified into similar (1) or dissimilar (-1). We use the following cosine similarity as the similarity metric:

$$y = \cos(\tilde{Z}_1, \tilde{Z}_2). \tag{9}$$

We evaluate the model performance by comparing predicted similarity scores with ground truth, and detail the performance of our proposed model on these two types of tasks in subsequent sections.

4 Experiments

In this section, we systematically evaluate our proposed model on GED and BSD tasks with eight benchmark datasets. Specifically, we first introduce the datasets for these two mentioned tasks, and then describe the baselines that are to be compared with our model, and the details of the experimental setup. We finally present the main evaluation results and perform the ablation experiment.

4.1 Datasets

We evaluate our model on eight public datasets, which are commonly used for graph similarity problem [Ling *et al.*, 2021]. The statistics of datasets are summarized in Table 1. In the GED task, we employ two benchmark datasets, including Aids700nef and Linux1000. We followed the previous works [Ling *et al.*, 2021] and divide each dataset into 60%, 20%, 20%, for training, validation, and testing set respectively. OpenSSL (OS) and FFmpeg (FF) datasets are employed for the BSD task, where [3, 200] means the size ranges of pairs of graphs. The datasets are divided into 10%, 10%, 80% for training/ validation/ testing set for supervised methods and 80%, 10%, 10% for unsupervised methods.

4.2 Baselines & Settings

We compare CGMN with two supervised GNN methods and three state-of-the-art unsupervised GNN models. Supervised methods include GCN [Kipf and Welling, 2017] and GIN [Xu *et al.*, 2019]. Unsupervised methods include DGI [Velickovic *et al.*, 2019], GRACE [Zhu *et al.*, 2020], and ScGSLC [Li *et al.*, 2021]. The first four are graph representation learning methods, and the last one is an unsupervised clustering-based graph similarity learning method.

We employ a 3-layers GCN as our core encoder. The learning rate is uniformly set as 0.0001 and the latent dimension of graph encoders, projectors, and the predictor are fixed to 100. We regard the GED task and BSD task as downstream tasks. In the GED task, we use the same settings for the supervised baselines, including dataset partitioning, learning rate, etc. We train the supervised models using mean square error (MSE) loss with all the ground truth from the training set.

Datasets	Graphs	AvgN	AvgE	Classes
Aids700nef	700	8.90	8.80	-
Linux1000	1000	7.58	6.94	-
OS [3, 200]	73,953	15.73	21.97	4,249
OS [20, 200]	15,800	44.89	67.15	1,073
OS [50, 200]	4,308	83.68	127.75	338
FF [3, 200]	83,008	18.83	27.02	10,376
FF [20, 200]	31,696	51.02	75.88	7,668
FF [50, 200]	10,824	90.93	136.83	3,178

Table 1: Statistics of the datasets.

For unsupervised baselines, we use a multi-layer perceptron (MLP) to map the predicted similarity scores to GED scores distribution intervals, for which we fine-tune the mapping between the predicted scores and the GED scores by using a MSE loss with 1% GED ground truth. In the BSD task, we train supervised models using the same approaches as GED. For unsupervised baselines, we use their self-contained losses for graph representation learning. We repeat all experiments ten times and report the average score and standard deviation.

4.3 Evaluation Results

We systematically evaluate the overall performance on the two tasks, the ablation study, and the parameter analysis.

GED task. We mainly use MSE as an evaluation metric of the GED task, which measures the mean squared error between the predicted similarity scores and the GED ground truth. We also provide other evaluation metrics, such as Spearman's Rank Correlation Coefficient (ρ) and Kendall's Rank Correlation Coefficient (τ) to comprehensively measure how well the predicted scores match the true scores. Precision at k ($p@k$) is calculated by dividing the intersection of the predicted top k results and the ground truth top k results. As shown in Table 2, our model CGMN outperforms other baseline methods in four metrics on Aids700nef and even partially surpasses the supervised algorithm. Compared to the optimal supervised models, the MSE value of CGMN has been increased by 28.43%, and compared with the unsupervised algorithms, the MSE value increases by 45.46%. On Linux1000 dataset, we achieve optimality in three metrics and suboptimality in the other two evaluation metrics. Compared with the supervised models and the unsupervised models, our model CGMN improves the value of MSE by 12.28% and 21.67% respectively, which demonstrates the superiority of our new model.

BSD task. We use the area under the ROC curve (AUC) as an evaluation metric for the BSD task, where AUC and model performance are positively correlated. The overall results are presented in Table 3. We bold the optimal results and '$\pm$' indicates a numerical range. Compared to unsupervised methods, our model CGMN consistently achieves the superior performance in terms of all metrics on all six datasets. CGMN can learn a good embedding function, which can be generalized to invisible test plots.

Ablation study. To verify the effectiveness of cross-view interaction and cross-graph interaction, we employ two

Datasets	Methods	MSE (10^{-3})	ρ	τ	p@10	p@20
Aids700nef	GCN	11.395±1.315	0.577±0.021	0.418±0.018	0.041±0.002	0.077±0.003
	GIN	9.280±0.163	0.629±0.020	0.462±0.016	0.044±0.018	0.096±0.021
	DGI	15.009±0.347	0.231±0.093	0.164±0.061	0.039±0.006	0.076±0.001
	GRACE	12.176±1.693	0.366±0.186	0.261±0.134	0.038±0.004	0.072±0.018
	ScGSLC	13.060±0.193	0.394±0.133	0.281±0.097	0.080±0.026	**0.142±0.044**
	CGMN	**6.641±2.227**	**0.674±0.129**	**0.502±0.107**	**0.084±0.019**	0.140±0.024
Linux1000	GCN	11.986±1.532	0.569±0.033	0.411±0.028	0.043±0.005	0.071±0.001
	GIN	22.188±5.259	0.647±0.112	0.484±0.099	0.081±0.018	0.084±0.025
	DGI	33.854±0.013	0.052±0.018	0.039±0.002	0.035±0.020	0.073±0.016
	GRACE	14.180±2.080	0.852±0.019	0.673±0.025	**0.443±0.155**	**0.452±0.175**
	ScGSLC	13.423±2.038	0.840±0.010	0.658±0.021	0.192±0.095	0.213±0.120
	CGMN	**10.514±1.178**	**0.873±0.013**	**0.700±0.015**	0.307±0.071	0.330±0.091

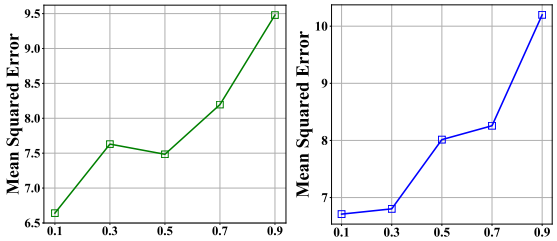
Table 2: Experimental results on the GED datasets in terms of five evaluation metrics.

Methods	OS [50, 200]	OS [20, 200]	OS [3, 200]	FF [50, 200]	FF [20, 200]	FF [3, 200]
GCN	67.24±1.14	68.09±1.01	73.51±0.72	78.41±0.49	79.47±0.08	80.88±0.18
GIN	66.60±0.10	63.85±0.56	75.65±0.30	78.38±0.20	81.25±0.57	**81.82±0.25**
DGI	67.55±2.76	63.58±1.96	72.58±2.36	86.10±0.66	80.82±2.22	66.28±0.30
GRACE	68.84±2.45	67.01±0.49	69.86±0.29	85.44±0.27	75.05±0.73	66.95±2.78
ScGSLC	67.43±0.82	61.46±0.33	63.28±0.09	**87.57±0.82**	83.27±0.71	69.80±1.22
CGMN	**80.89±0.20**	**78.15±0.85**	**75.94±1.86**	86.11±0.98	**86.76±0.85**	77.98±2.69

Table 3: Experimental results on the BSD datasets in terms of AUC scores (%).

Methods	MSE	ρ	τ	p@10	p@20
CGMN w/o cross-view	8.239	0.614	0.451	0.064	0.114
CGMN w/o cross-graph	8.753	0.537	0.387	0.050	0.091
CGMN	6.641	0.674	0.502	0.084	0.140

Table 4: Ablation study on Aids700nef.



(a) Masking node features. (b) Removing edges.

Figure 5: Influence on parameters.

CGMN variants on Aids700nef, each of which has one of the key components removed. Table 4 shows the results of the ablation study, where w/o represents the removal of a strategy. Experimental results show that the performance of our model degrades without one of the key strategies on Aids700nef. We find that the CGMN achieves 24.13% higher relative to the others variants on MSE (10^{-3}). Among other evaluation metrics, CGMN also achieves the best performance compared to its two variants, which proves the effectiveness of the combination of our two proposed schemes.

Parameter sensitivity analysis. We analyse two important parameters, i.e., the probability of masking features and the probability of removing edges in generating two views during graph augmentation. We carry out experiments on the Aids700nef dataset with different parameters as an example, and Fig. 5 shows the effect of parameter sensitiveness on the CGMN performance. The horizontal coordinates represent the probability of masking node features or removing edges, and the vertical coordinates represent the MSE (10^{-3}). For example, 0.1 means randomly masking node features with probability 10%. In Fig. 5 (a) and (b), despite the fluctuations in the line chart, the overall trend of MSE values is increasing. This means that increasing the modification probability excessively may distort the graph structures and features information, resulting in significant performance degradation.

5 Conclusion

Graph similarity learning is pivotal in graph problems. In this paper, we propose a novel unsupervised graph similarity learning method CGMN. Using a siamese GNN as the backbone, we design a cross-view interaction strategy to enrich the self-supervised signal. To further match nodes across graphs, we introduce a cross-graph interaction mechanism to conduct cross-graph matching between nodes. The experimental results demonstrate the superiority and effectiveness of the new approach compared with the SOTA methods.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under grants 61876128 and Meituan Project.

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. In *ICDM*, pages 384–392, 2019.

[Bai *et al.*, 2021] Yunsheng Bai, Derek Xu, Yizhou Sun, and Wei Wang. Glsearch: Maximum common subgraph detection via learning to search. In *ICML*, 2021.

[Bunke, 1997] Horst Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607. PMLR, 2020.

[Hassani and Khasahmadi, 2020] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, pages 4116–4126. PMLR, 2020.

[He *et al.*, 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.

[Hjelm *et al.*, 2019] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[Jia *et al.*, 2021] Renqi Jia, Xiaofei Zhou, Linhua Dong, and Shirui Pan. Hypergraph convolutional network for group recommendation. In *ICDM*, pages 260–269. IEEE, 2021.

[Jin *et al.*, 2021a] Di Jin, Cuiying Huo, Chundong Liang, and Liang Yang. Heterogeneous graph neural network via attribute completion. In *WWW*, pages 391–400, 2021.

[Jin *et al.*, 2021b] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip Yu, and Weixiong Zhang. A survey of community detection approaches: From statistical modeling to deep learning. *TKDE*, DOI: 10.1109/TKDE.2021.3104155, 2021.

[Jin *et al.*, 2021c] Ming Jin, Yizhen Zheng, Yuan-Fang Li, Chen Gong, Chuan Zhou, and Shirui Pan. Multi-scale contrastive siamese networks for self-supervised graph representation learning. In *IJCAI*, pages 1477–1483, 2021.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Li *et al.*, 2019] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, pages 3835–3845. PMLR, 2019.

[Li *et al.*, 2021] Junyi Li, Wei Jiang, Henry Han, Jing Liu, Bo Liu, and Yadong Wang. Scgslc: An unsupervised graph similarity learning framework for single-cell rna-seq data clustering. *Computational Biology and Chemistry*, 90:107415, 2021.

[Ling *et al.*, 2021] Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Alex X. Liu, Chunming Wu, and Shouling Ji. Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021.

[Neuhaus *et al.*, 2006] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *S+SSPR*, pages 163–172. Springer, 2006.

[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.

[Sanfeliu and Fu, 1983] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

[Su *et al.*, 2021] Yixin Su, Rui Zhang, Sarah M. Erfani, and Junhao Gan. Neural graph matching based collaborative filtering. In *SIGIR*, pages 849–858. ACM, 2021.

[Velickovic *et al.*, 2019] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

[Wan *et al.*, 2021] Sheng Wan, Shirui Pan, Ping Zhong, Xiaojun Chang, Jian Yang, and Chen Gong. Dual interactive graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2021.

[Wang *et al.*, 2020] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning. In *NeurIPS*, 2020.

[Wang *et al.*, 2021] Runzhong Wang, Tianqi Zhang, Tianshu Yu, Junchi Yan, and Xiaokang Yang. Combinatorial learning of graph edit distance via dynamic embedding. In *CVPR*, pages 5241–5250, 2021.

[Xu *et al.*, 2017] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. Neural network-based graph embedding for cross-platform binary code similarity detection. In *CCS*, pages 363–376, 2017.

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

[Yu *et al.*, 2021] Zhizhi Yu, Di Jin, Ziyang Liu, Dongxiao He, Xiao Wang, Hanghang Tong, and Jiawei Han. As-gcn: Adaptive semantic architecture of graph convolutional networks for text-rich networks. In *ICDM*, 2021.

[Zhang *et al.*, 2021] Zhen Zhang, Jiajun Bu, Martin Ester, Zhao Li, Chengwei Yao, Zhi Yu, and Can Wang. H2mn: Graph similarity learning with hierarchical hypergraph matching networks. In *SIGKDD*, pages 2274–2284, 2021.

[Zhu *et al.*, 2020] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *ICML*, 2020.