Data Augmentation for Graph Classification

Jiajun Zhou
Zhejiang University of Technology
HangZhou, China
jjzhou@zjut.edu.cn

Jie Shen
Zhejiang University of Technology
HangZhou, China
shenj@zjut.edu.cn

Qi Xuan
Zhejiang University of Technology
HangZhou, China
xuanqi@zjut.edu.cn

ABSTRACT

Graph classification, which aims to identify the category labels of graphs, plays a significant role in drug classification, toxicity detection, protein analysis etc. However, the limitation of scale of benchmark datasets makes it easy for graph classification models to fall into over-fitting and undergeneralization. Towards this, we introduce data augmentation on graphs and present two heuristic algorithms: *random mapping* and *motif-similarity mapping*, to generate more weakly labeled data for small-scale benchmark datasets via heuristic modification of graph structures. Furthermore, we propose a generic model evolution framework, named *M-Evolve*, which combines graph augmentation, data filtration and model retraining to optimize pre-trained graph classifiers. Experiments conducted on six benchmark datasets demonstrate that *M-Evolve* helps existing graph classification models alleviate over-fitting when training on small-scale benchmark datasets and yields an average improvement of 3–12% accuracy on graph classification tasks.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; • **Computing methodologies → Supervised learning by classification**.

KEYWORDS

Graph Classification; Data Augmentation; Model Evolution

1 INTRODUCTION

Graph classification, or network classification, has recently attracted considerable attention from different fields like bioinformatics [1] and chemoinformatics [3]. For instance, in bioinformatics, proteins or enzymes can be represented as labeled graphs, in which vertices are atoms and edges represent chemical bonds that connect atoms. The task of graph classification is to classify these molecular graphs according to their chemical properties like carcinogenicity, mutagenicity and toxicity.

However, in bioinformatics and chemoinformatics, the scale of the known benchmark graph datasets is generally in the range of tens to thousands, which is far from the scale of real-world social network datasets like COLLAB and IMDB [8]. Despite the advances of various graph classification methods, from graph kernels, graph embedding to graph neural networks, the limitation of data scale makes them easily fall into the dilemmas of over-fitting and undergeneralization.

To solve the above problem, we take an effective approach to study data augmentation on graphs and develop two graph augmentation methods, called *random mapping* and *motif-similarity mapping*, respectively. The idea is to generate more virtual data for small datasets via heuristic modification of graph structures. Since the generated graphs are artificial and treated as weakly labeled data, their availability remains to be verified. Therefore, we introduce a concept of label reliability, which reflects the matching degree between examples and their labels, to filter fine augmented examples from generated data. Furthermore, we introduce a model evolution framework, named *M-Evolve*, which combines graph augmentation, data filtration and model retraining to optimize classifiers. We demonstrate that *M-Evolve* achieves a significant improvement of performance on graph classification.

The main contributions of our work are summarized as follows:

- We effectively utilize the technique of data augmentation on graph classification, and develop two methods to generate effective weakly labeled data for graph benchmark datasets.
- We propose a generic model evolution framework named *M-Evolve* for enhancing graph classification, which can be easily combined with existing graph classification models.
- We conduct experiments on six benchmark datasets. Experimental results demonstrate the superiority of *M-Evolve* in helping five graph classification algorithms to achieve significant improvement of performances.

2 METHODOLOGY

Let $G = (V, E)$ be an undirected and unweighted graph, which consists of a vertex set $V = \{v_i \mid i = 1, \ldots, n\}$ and an edge set $E = \{e_i \mid i = 1, \ldots, m\}$. The topological structure of graph G is represented by an $n \times n$ adjacency matrix A with $A_{ij} = 1$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise. Dataset that contains a series of graphs is denoted as $D = \{(G_i, y_i) \mid i = 1, \ldots, t\}$, where y_i is the label of graph G_i. For D, an upfront split will be applied to yield disjoint training, validation and testing set, denoted as D_{train}, D_{val} and D_{test}, respectively. The original classifier C will be pre-trained on D_{train} and D_{val}.

Problem Definition: We explore data augmentation technique for graph classification problem with heuristic paradigm and consider
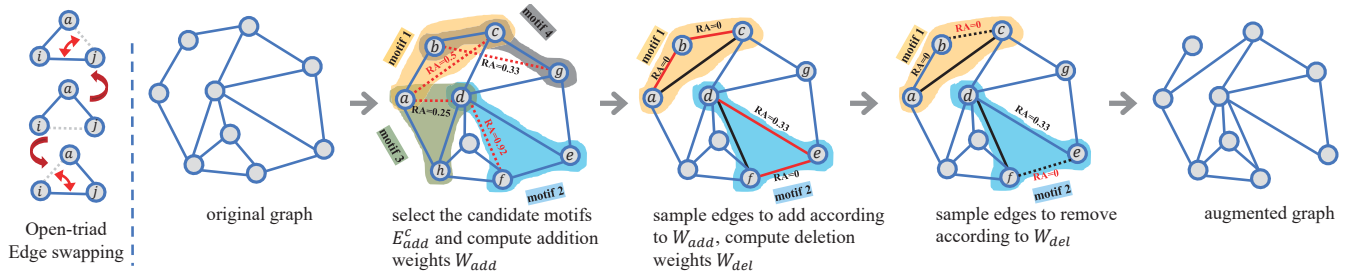
Figure 1: Schematic depiction of graph augmentation. *Left*: Open-triad motif and heuristic edge swapping. *Right*: An example for graph augmentation via motif-similarity mapping; red lines is the candidates and black lines is the modified edges.

optimizing graph classifier. Specifically, we aim to update a classifier with augmented data, which are first generated via graph augmentation and then filtered in terms of their label reliability. During graph augmentation, our purpose is to map the graph $G \in D_{train}$ to a new graph G' with the formal format: $f : (G, y) \mapsto (G', y)$. We treat the generated graphs as weakly labeled data and classify them into two groups via a label reliability threshold θ learnt from D_{val}. Then the augmented set D'_{train} filtered from generated graph pool D_{pool} will be merged with D_{train} to produce the training set:

$$D^{new}_{train} = D_{train} + D'_{train}, \quad D'_{train} \subset D_{pool}. \quad (1)$$

Finally, we finetune or retrain the classifier with D^{new}_{train}, and evaluate it on the testing set D_{test}.

2.1 Graph Augmentation

Graph augmentation aims to expand training data via artificially creating more reasonable virtual data from a limited set of graphs. In this paper, we consider augmentation as a topological mapping, which is conducted via heuristic modification of graph structure. In order to ensure the approximate reasonability of the generated virtual data, our graph augmentation will follow these principles: 1) edge modification, where G' is a partially modified graph with some of the edges added/removed from G; 2) structure property preservation, where augmentation operation keeps the graph connectivity and the number of edges constant. During edge modification, those edges removed from graph are sampled from the candidate edge set E^c_{del}, while the edges added to graph are sampled from the candidate pairwise vertices set E^c_{add}. The construction of candidate sets varies for different methods, as further discussed below.

2.1.1 Random Mapping.
Here, consider *random mapping* as a simple baseline. The candidate sets are constructed as follows:

$$E^c_{del} = E, \quad E^c_{add} = \{(v_i, v_j) \mid A_{ij} = 0; \ i \neq j\}. \quad (2)$$

Notably, in random scenario, E^c_{del} is actually the edge set of graph, and E^c_{add} is the set of virtual edges which consist of unlinked pairwise vertices. Then, one can get the set of edges added/removed from G via sampling from the candidate sets randomly:

$$E_{del} = \{e_i \mid i = 1, \ldots, \lceil m \cdot \beta \rceil\} \subset E^c_{del},$$
$$E_{add} = \{e_i \mid i = 1, \ldots, \lceil m \cdot \beta \rceil\} \subset E^c_{add}, \quad (3)$$

where β is the budget of edge modification and $\lceil x \rceil = \mathbf{ceil}(x)$. Finally, based on the *random mapping*, the connectivity structure of the original graph is modified to generate a new graph:

$$G' = (V, (E \cup E_{add}) \backslash E_{del}). \quad (4)$$

2.1.2 Motif-Similarity Mapping.
Graph motifs are sub-graphs that repeat themselves in a specific graph or even among various graphs. Each of these sub-graphs, defined by a particular pattern of interactions between vertices, may describe a framework in which particular functions are achieved efficiently. In this paper, we just consider open-triad motifs with chain structures. As shown in the left of Figure 1, open-triad $\wedge^a_{ij}$ is equivalent to length-2 paths emanating from the head vertex v_i that induce a triangle.

The *motif-similarity mapping* aims to finetune these motifs to approximately equivalent ones via edge swapping. During edge swapping, edge addition takes effect between the head and the tail vertices of the motif, while edge deletion removes an edge in the motif via weighted random sampling. For all open-triad motifs $\wedge_{ij}$ which has head vertex v_i and tail vertex v_j, the candidate pairwise vertices set is denoted as:

$$E^c_{add} = \{(v_i, v_j) \mid A_{ij} = 0, A^2_{ij} \neq 0; i \neq j\}. \quad (5)$$

Then one can get E_{add}, the set of edges added to G, via weighted random sampling from E^c_{add}. For each $\wedge_{ij}$ involving pairwise vertices (v_i, v_j) in E_{add}, we remove one edge from it via weighted random sampling and all of these removed edges constitute E_{del}.

Notably, we assigns all entries in E^c_{add} and $\wedge_{ij}$ with relative sampling weights which are associated with the vertex similarity scores. Specifically, before sampling, we compute the similarity scores over all entries in E^c_{add} using *Resource Allocation (RA)* index which has been proven its superiority among several local similarity indices in [10]. For each entry (v_i, v_j) in E^c_{add}, the RA score s_{ij} and addition weight w^{add}_{ij} can be computed as follows:

$$s_{ij} = \sum_{z \in \Gamma(i) \cap \Gamma(j)} \frac{1}{d_z}, \ S = \{s_{ij} \mid \forall (v_i, v_j) \in E^c_{add}\},$$
$$w^{add}_{ij} = \frac{s_{ij}}{\sum_{s \in S} s}, \ W_{add} = \{w^{add}_{ij} \mid \forall (v_i, v_j) \in E^c_{add}\}, \quad (6)$$

where $\Gamma(i)$ denotes the one-hop neighbors of v_i and d_z denotes the degree of vertex z. Weighted random sampling means that the probability for an entry in E^c_{add} to be selected is proportional to its addition weight w^{add}_{ij}. Similarly, during edge deletion, the probability of edge sampled from $\wedge_{ij}$ is proportional to the deletion weight w^{del}_{ij} as follows:

$$w^{del}_{ij} = 1 - \frac{s_{ij}}{\sum_{s \in S} s}, \ W_{del} = \{w^{del}_{ij} \mid \forall (v_i, v_j) \in \wedge_{ij}\}, \quad (7)$$

which means that these edges with smaller RA scores have more chance to be removed. Finally, the augmented graph can be obtained via Eq. (4). It is worth noting that many other similarity indices such as CN and Katz [10] can also be applied into this scheme.

2.2 Model Evolution

2.2.1 **Data Filtration**. Due to the topology dependency of graph structured data, the examples generated via graph augmentation may lose original semantics. By assigning the label of the original graph to the generated graph directly during graph augmentation, one cannot determine whether the assigned label is reliable. Therefore, the concept of label reliability is employed here to measure the matching degree between examples and labels.

Each graph G_i in D_{val} will be fed into classifier C to obtain the prediction vector $\mathbf{p}_i \in \mathbb{R}^{|Y|}$, which represents the probability distribution how likely an input example belongs to each possible class. $|Y|$ is the number of classes for labels. Then a probability confusion matrix $\mathbf{Q} \in \mathbb{R}^{|Y| \times |Y|}$, in which the entry q_{ij} represents the average probability that the classifier classifies the graphs of i-th class into j-th class, is computed as follows:

$$\mathbf{q}_k = \frac{1}{\Omega_k} \sum_{y_i = k} \mathbf{p}_i , \quad \mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{|Y|}] , \tag{8}$$

where Ω_k is the number of graphs belonging to k-th class in D_{val} and $\mathbf{q}_k$ is the average probability distribution of k-th class.

The label reliability of an example (G_i, y_i) is defined as the product of example probability distribution $\mathbf{p}_i$ and class probability distribution $\mathbf{q}_{y_i}$ as follows:

$$r_i = \mathbf{p}_i^\top \mathbf{q}_{y_i} . \tag{9}$$

A threshold θ used to filter the generated data is defined as:

$$\theta = \arg\min_\theta \sum_{(G_i, y_i) \in D_{val}} \Phi[(\theta - r_i) \cdot g(G_i, y_i)] , \tag{10}$$

where $g(G_i, y_i) = 1$ if $C(G_i) = y_i$ and $g(G_i, y_i) = -1$ otherwise, and $\Phi(x) = 1$ if $x > 0$ and $\Phi(x) = 0$ otherwise.

2.2.2 **Framework**. Model evolution aims to optimize classifiers via graph augmentation, data filtration and model retraining iteratively, and ultimately improve the performance on graph classification task. The procedure of *M-Evolve* is shown in Algorithm 1.

3 EVALUATION

3.1 Experimental Setup

3.1.1 **Data**. We evaluate our methods on six benchmark datasets: Mutag, PTC-MR, ENZYMES, KKI, Peking-1 and OHSU [4]. The specifications of these datasets are given in Table 1, where *bias* is the proportion of the dominant class.

3.1.2 **Graph Classification Methods.** Five graph classification methods, i.e., SF [2], Graph2Vec [5], NetLSD [6], Gl2Vec [7], Diffpool [9], are used in the experiments. The first two are graph embedding, the middle two are kernel models, and the last one is GNN model. For all graph kernel and embedding methods, we implement graph classification by using the following machine learning classifiers: SVM based on radial basis kernel (SVM), Logistic Regression (Log), k-Nearest Neighbors (KNN) and Random Forest (RF).

3.1.3 **Parameter Settings.** Each dataset is split into training, validation and testing sets with a proportion of 7:1:2. We repeat 5-fold cross validation 10 times and report the average accuracy across all trials. For all kernel and embedding methods, the feature dimension is set to 128. We set the budget of edge modification β as 0.15. Furthermore, the evolution iterations T is set to 5.

Algorithm 1: *M-Evolve*

Input: Training set D_{train}, validation set D_{val}, graph augmentation function f, evolution iterations T.
Output: Evolutionary model C'.

1 Pre-training classifier C using D_{train} and D_{val} ;
2 Initalize *iteration* = 0;
3 **for** *iteration* < T **do**
4   Graph augmentation: $D_{pool} \leftarrow f(D_{train})$;
5   For all graphs G_i in D_{val} classified by C, get $\mathbf{p}_i$;
6   Compute probability confusion matrix $\mathbf{Q}$ via Eq. 8 ;
7   For all graphs G_i in D_{val} classified by C, get r_i via Eq. 9;
8   Compute the label reliability threshold θ via Eq. 10 ;
9   For all examples (G_i, y_i) in D_{pool} classified by C, compute r_i, **if** $r_i > \theta$, $D_{train}.append((G_i, y_i))$;
10   Get the evolutionary classifier: $C' \leftarrow \text{retrain}(C, D_{train})$;
11   *iteration* $\leftarrow$ *iteration* + 1 ;
12      $C \leftarrow C'$;
13 **end** ;
14 **return** C';

Table 1: Dataset properties.

Collections	Dataset	$|D|$	$|Y|$	*Avg.*$|V|$	*Avg.*$|E|$	*bias* (%)
Chemical Compounds	MUTAG	188	2	17.93	19.79	66.5
	PTC-MR	344	2	14.29	14.69	55.8
	ENZYMES	600	6	32.63	62.14	16.7
Brain	KKI	83	2	26.96	48.42	55.4
	Peking-1	85	2	39.31	77.35	57.6
	OHSU	79	2	82.01	199.66	55.7

3.2 Enhancement for Graph Classification

Table 2 reports the results of performance comparison between evolutionary models and original models, from which we can see that there is a significant boost in classification performance across all six datasets. Overall, these models combined with the proposed *M-Evolve* framework obtain higher average classification accuracy in most cases and the *M-Evolve* achieves a 97.06% success rate on the enhancement of graph classification. Moreover, the far-right column gives the average relative improvement rate (Avg RIMP) in accuracy, from which we can see that the *M-Evolve* combined with *motif-similarity mapping* obtains the best results overall. As a reasonable explanation, similarity mechanism tends to link vertices with higher similarity and is capable of optimizing topological structure legitimately, while the motif mechanism achieves edge modification via local edge swapping, which has less effect on the degree distribution of the graph.

Furthermore, we visualize the data distribution and the decision boundary of models before and after model evolution, as shown in Figure 2, to investigate how the *M-Evolve* framework achieves interpretable enhancement of graph classification. Due to space limit, we only present the visualization results of graph classification based on these combinations (*SF* + {*SVM*, *KNN*}). As we can see, there is a significant increase in the scale of dataset, indicating that graph augmentation effectively enriches the training data and the new data distribution is more conducive to the training

Table 2: Graph classification results of original and evolutionary model.

| Dataset	Mapping	Graph Classification Model																Diffpool	Avg RIMP
		SF				NetLSD				Graph2Vec				Gl2Vec					
		SVM	Log	KNN	RF	SVM	Log	KNN	RF	SVM	Log	KNN	RF	SVM	Log	KNN	RF		
MUTAG	original	0.822	0.824	0.824	0.846	0.823	0.829	0.828	0.836	0.737	0.820	0.784	0.820	0.746	0.830	0.800	0.817	0.801	–
	random	0.853	0.844	0.835	0.878	0.855	0.851	0.853	0.886	0.756	**0.854**	0.790	**0.844**	0.748	0.842	0.820	0.840	0.810	2.67%
	motif-similarity	**0.863**	**0.849**	**0.838**	**0.890**	**0.860**	**0.864**	**0.858**	**0.892**	**0.759**	0.849	**0.806**	0.842	**0.762**	**0.848**	**0.829**	**0.846**	**0.831**	**3.60%**
PTC-MR	original	0.551	0.566	0.577	0.587	0.543	0.578	0.548	0.576	0.571	0.518	0.509	0.549	0.572	0.538	0.507	0.550	0.609	–
	random	0.611	**0.595**	**0.605**	0.617	0.579	0.580	0.590	0.607	0.580	0.572	0.547	0.592	0.587	0.571	0.527	0.594	**0.639**	5.82%
	motif-similarity	**0.613**	0.589	0.601	**0.624**	**0.581**	**0.581**	**0.597**	**0.620**	**0.590**	**0.579**	**0.553**	**0.593**	**0.587**	**0.579**	**0.545**	**0.602**	0.630	**6.60%**
ENZYMES	original	0.309	0.393	0.287	0.397	0.337	**0.248**	0.304	0.349	0.361	0.253	0.283	0.337	**0.348**	0.268	0.238	0.318	0.487	–
	random	0.347	0.412	0.302	0.412	0.351	0.237	0.327	0.369	0.336	0.269	**0.290**	0.346	0.286	0.273	0.259	0.334	0.500	2.60%
	motif-similarity	**0.363**	**0.414**	**0.317**	**0.414**	**0.375**	0.248	**0.334**	**0.376**	0.352	**0.270**	0.289	**0.352**	0.291	**0.280**	**0.260**	**0.339**	**0.506**	**5.00%**
KKI	original	0.550	0.500	0.520	0.517	0.548	0.524	0.512	0.496	0.549	0.527	0.524	0.552	0.538	0.502	0.526	0.502	0.523	–
	random	**0.606**	0.544	0.554	0.622	0.599	0.535	0.553	0.562	0.580	0.568	0.594	0.574	0.556	0.508	0.544	0.544	0.586	8.10%
	motif-similarity	0.605	**0.559**	**0.561**	**0.649**	**0.618**	**0.550**	**0.558**	**0.582**	**0.587**	**0.590**	**0.603**	**0.632**	**0.574**	**0.524**	**0.597**	**0.582**	**0.612**	**12.07%**
Peking-1	original	0.578	0.548	0.541	0.558	0.605	0.612	0.589	0.591	0.572	0.522	0.474	0.522	0.555	0.522	0.521	0.521	0.586	–
	random	0.660	**0.592**	0.603	0.627	0.652	**0.631**	0.662	0.654	0.579	0.536	**0.546**	0.576	0.584	0.555	**0.569**	0.607	**0.631**	9.08%
	motif-similarity	**0.670**	0.583	**0.624**	**0.663**	**0.694**	0.627	**0.671**	**0.699**	**0.581**	**0.565**	0.539	**0.630**	**0.607**	**0.563**	0.562	**0.625**	0.626	**11.88%**
OHSU	original	0.610	0.595	0.610	0.667	0.547	0.489	0.549	0.581	**0.557**	0.577	0.585	0.567	0.557	0.541	0.544	0.557	0.543	–
	random	**0.663**	0.635	**0.644**	0.687	0.575	0.494	0.582	**0.641**	0.557	0.620	0.602	0.645	0.564	0.595	0.625	0.610	0.600	6.87%
	motif-similarity	0.656	**0.640**	0.636	**0.707**	**0.638**	**0.501**	**0.587**	0.638	0.557	**0.625**	**0.633**	**0.650**	**0.572**	**0.605**	**0.625**	**0.652**	**0.604**	**8.83%**
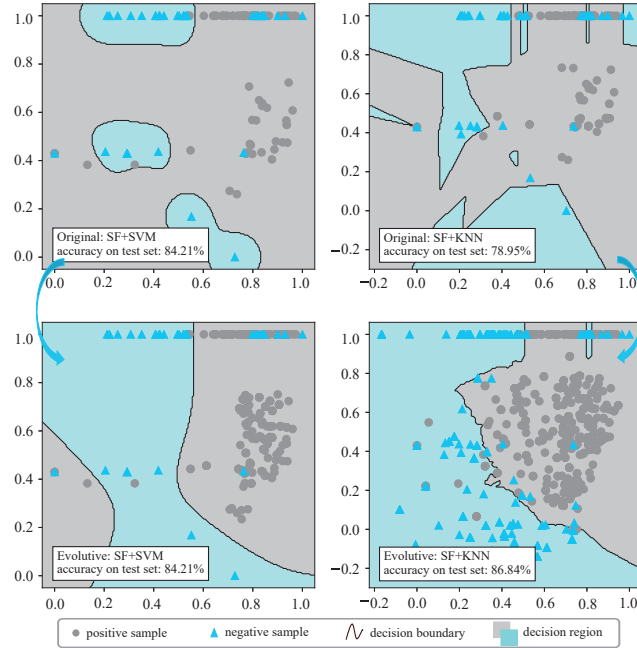
Figure 2: Visualization of training data distribution and decision boundaries of graph classifiers on MUTAG dataset.

of classifiers. Moreover, the decision regions of the non-dominant class are fragmented and scattered in the original models. During model evolution, scattered regions tend to merge, and the original decision boundaries are optimized to smoother ones.

In summary, graph augmentation can efficiently increase the data scale, indicating its ability in enriching data distribution. And the entire *M-Evolve* framework is capable of optimizing the decision boundaries of the classifiers and ultimately improving their generalization performances.

4 CONCLUSION

In this paper, we introduce a concept of graph augmentation in graph structured data and present two heuristic algorithms to generate weakly labeled data for small-scale benchmark datasets via heuristic transformation of graph structure. Furthermore, we propose a generic model evolution framework that combines graph augmentation, data filtration and model retraining to optimize pretrained graph classifiers. Experiments conducted on six benchmark datasets demonstrate that our proposed framework behaves surprisingly well and helps existing graph classification models alleviate over-fitting when training on small-scale benchmark datasets and achieve significant improvement of classification performance.

ACKNOWLEDGMENTS

REFERENCES

[1] 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.
[2] Nathan de Lara and Edouard Pineau. 2018. A simple baseline algorithm for graph classification. *arXiv preprint arXiv:1810.09155* (2018).
[3] 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.
[4] Kristian Kersting, Nils M Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. 2016. Benchmark data sets for graph kernels, 2016. *URL http://graphkernels. cs. tu-dortmund. de* 795 (2016).
[5] 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).
[6] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. 2018. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2347–2356.
[7] Kun Tu, Jian Li, Don Towsley, Dave Braines, and Liam D Turner. 2019. gl2vec: Learning feature representation using graphlets for directed networks. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 216–221.
[8] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1365–1374.
[9] 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*. 4800–4810.
[10] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. 2009. Predicting missing links via local information. *The European Physical Journal B* 71, 4 (2009), 623–630.