Knowledge-Based Systems 230 (2021) 107383

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Learning graph-level representation from local-structural distribution with Graph Neural Networks

Wei-Xiang Sun, Hui Xue *

School of Computer Science and Engineering, Southeast University, Nanjing, China
MOE Key Laboratory of Computer Network and Information Integration (Southeast University), China

ARTICLE INFO

Article history:
Received 31 January 2021
Received in revised form 4 August 2021
Accepted 10 August 2021
Available online 12 August 2021

Keywords:
Graph-level representation learning
Local-structural distribution
Graph Neural Networks

ABSTRACT

Graph Neural Networks has been proved to be successful in learning the latent vector representation for local structures. While applying this technique to some real-world tasks, such as property prediction for molecules, a graph-level representation generator should be required to integrate all these local-structural embeddings. Limited by varying graph scale and the absence of node order, existing graph-level representation generation approaches usually adopt rough ways, *e.g.*, compressing all local-structural embeddings into a single vector, which cannot simultaneously retain both the local- and global-structural information in graph-level representations. To address this problem, this paper introduces a novel and more powerful approach to learn graph-level representation from local-structural distribution. Firstly, our approach employs a batch strategy to discretize the embeddings of local structures by their structural semantics, which can provide interactive information to approximately align the local-structural embeddings for varying graphs. According to Wasserstein metric, the aligned structural information is beneficial to capture the similarity among graphs. Then, our approach further integrates these aligned local-structural embeddings to construct a resolution-controllable structural histogram as the graph-level representation. Without over-compression, more local-structural information can be preserved. The experimental results show that our approach substantially outperformed the baselines on a range of real-world datasets in both graph classification and regression tasks.

1. Introduction

Graph-structured data is ubiquitous in many domains, such as biomedical science, social network analysis, and recommender systems. A graph can be used to represent almost any physical situation involving discrete objects and relationships among them [1]. For example, chemical compounds are usually described as the interaction graphs of the forming elements [2,3]; Social relationship networks can be expressed with complex topology [4]. Compared with the data represented in Euclidean space, such as sequence data and grid-like images, there are several challenges in the analysis of such graph-structured data in machine learning: 1. the conflict *between* the irregular graph topologies with variable scales *and* the fixed input shape of machine learning algorithms; 2. the absence of node order that aggravates the difficulty in aligning graph structures.

To address the problems above, amounts of graph representation learning algorithms have been proposed in the last decade of years. Most of them, such as DeepWalk [5], Node2Vec [6], Struc2Vec [7] are essentially designed to leverage the dependency among structural relationships to learn a latent representation for local structures. For example, DeepWalk and Node2Vec serialize the local-structural information around nodes by truncated random walk paths, then embed them via generalized language models. Very recently, inspired by the graph embedding methods and Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs) have been proposed and developed [8–12]. This technique follows an iterative scheme collectively, where each node aggregates the embeddings of its neighbors to compute its new embedding. Benefiting from the remarkably expressive ability and the end-to-end learning fashion, GNNs has been the most popular tool to analyze graph-structured data in machine learning [13].

Although GNNs can be applied to most node-level graph tasks naturally, in other real-world problems, such as property prediction of molecule [14] and community analysis of social networks [9,11], the local-structural embeddings learned by GNN have to be integrated into a global representation for each whole graph. To generate such graph-level representation, a collapsing-style operation (*e.g.*, global sum/average pooling) is commonly

* Corresponding author at: School of Computer Science and Engineering, Southeast University, Nanjing, China.
E-mail address: hxue@seu.edu.com (H. Xue).

https://doi.org/10.1016/j.knosys.2021.107383
0950-7051/

adopted [12,15,16]. In that, these local-structural embeddings are directly summarized into a single vector to represent the whole graph structure. This collapsing-style operation is inspired by traditional CNNs that are used widely in image data analysis [17,18]. However, considering the characteristic of graph data — in most cases, each graph node represents a real entity (*e.g.*, atom) rather than a useless pixel of background, therefore, graph data has much greater informational density than image data generally. Moreover, graphs are usually employed to describe complex compound systems (*e.g.*, electrical power systems) [19]. Consequently, this collapsing-style operation will lead to the over-compression of local-structural information and the loss of the interaction relationships among the components. Briefly, this way only focuses on global information but loses local-structural information. Zhang et al. have noticed this problem and proposed a different strategy, SortPool, to preserve more intact local-structural details [20]. Unfortunately, since SortPool just considers the structural order within a single graph when sorting, the natural alignment relationships of local structures are lost between different graphs. The non-alignment could cause feature shifting and rotating problems, so that puts forward higher requirements of learning shifting/rotating invariants for downstream networks. This will increase the difficulty in mining useful structural information, such as graph-level structural similarity, when solving downstream tasks. In addition, SortPool has to jettison some local-structural embeddings to unify the size of graph-level representations, which also hurts the perception of global-structural information. In other words, SortPool captures the local-structural details while losing the global outlook, which will be more likely to result in overfitting. Besides, attention mechanism [21] also has been introduced into the graph domain to capture the key local-structural information [22]. However, relevant studies also only consider the interactive information within a single graph, the alignment information between graphs cannot be captured as well. Furthermore, the expensive computation cost of the attention mechanism will limit its application, especially for large graphs.

In fact, for layer-wised neural network models that follow a serial architecture, any defective module of them may restrain the expressive power of the whole model. Accordingly, based on the investigations in the previous section, we know that even though GNNs have a powerful ability in learning structures, this ability may be restrained by the graph-level representation generation module when applying them to some graph-level tasks (e.g., graph classification/regression). To further highlight this view, we randomly initialized a graph neural network model for graph classification task and froze the parameters of its GNN layers. Only the graph-level representation generation module and classification network were optimized. The results are shown in Fig. 1. From the results of our approach, we can observe that the unoptimized GNN should show very competitive performance with the optimized one (see Table 1) for this task. However, the existing approaches seem not to tap such potential of the GNN. Thus, this phenomenon inspires us to consider that graph level representation generation module may be the bottleneck in graph level-representation learning using GNNs.

To break this bottleneck, this paper introduced a novel approach to generate graph-level representations by leveraging local-structural distributions. On the one hand, the local-structural distribution information is introduced into the graph-level representation generation process, which can effectively avoid the problem of losing the local-structural information due to the over-compression. On the other hand, the semantics of different structural embeddings are aligned by leveraging a batch alignment strategy and an alignment loss constraint. According to Wasserstein metric, we prove that the aligned structures are beneficial to the quantification of the distance between graph similarity. In conclusion, the main contributions of this paper can be summarized briefly as follows:

- This paper briefly dissects existing commonly-used graph-level representation generation strategies to better understand how they affect GNN to learn graph-level representation. We found that (1) Graph-level representation generation strategy has an obvious impact on the performance of GNN models in graph classification/regression tasks; (2) different strategies could focus on different structural properties; (3) Existing approaches cannot perfectly convey the structural information learned by GNN layers at both local and global levels.
- This paper proposes a novel approach that can construct graph-level representations with local-structural distributions. Different from existing methods, both local- and global-structural information can be conveyed for the downstream networks by our approach, which can further stimulate the latent capability of GNNs in structure learning. Interestingly, in some graph classification tasks, our approach can work well even using an unoptimized GNN (see Fig. 1).
- This paper develops a batch strategy to add interactive structural information into graph-level representations. This information is beneficial for local structures to recognize their own structural roles at the batch level, which makes it easy to align the local structures by their structural semantics for different graphs. This strategy lays a solid foundation to quantify the graph similarities by their local-structural distribution information.
- This paper designs experiments to verify the capability of related approaches and ours in structure learning at both local and global levels. The experimental results on graph classification and regression tasks show the significant superiority of our approach compared to the existing approaches.

2. Related work

2.1. Graph-level representation learning

Early related work can be traced back to 1965 when the Morgan algorithm [23] was designed to encode chemical structures in a discrete way. Later, Glen et al. refined this algorithm to provide a flexible graph descriptor [24], namely, circular fingerprint (CFP), which follows an iterative model that generates graph representation via a hash function to concatenate the features of the neighborhood in the previous iteration. Based on CFP, many kinds of improvements were proposed, such as ECFP [25] and NeuralFP [16]. However, these methods above mainly focus on graph-structured data analysis in biology and chemistry domains. In recent years, inspired by the success of convolutional neural networks (CNNs), a surge of more general graph representation learning models have been proposed. Some of them trim the graph data into a uniform shape that CNNs can accept, then the graph-level characteristics can be extracted by traditional CNN models [26,27], whereas others pursued a natural and intrinsical way to define the convolution operation on graph structure directly. Bruna et al. first proposed a spectral graph convolution network [28], where the convolution operation was defined in the Fourier domain, but matrix decomposition operations have to be applied in this algorithm. GCN [9] and fastGCN [29] solved such a problem of the expensive computation via fast local spectral-based convolutions. Over the same period, numerous scholars attempted to design graph convolutional

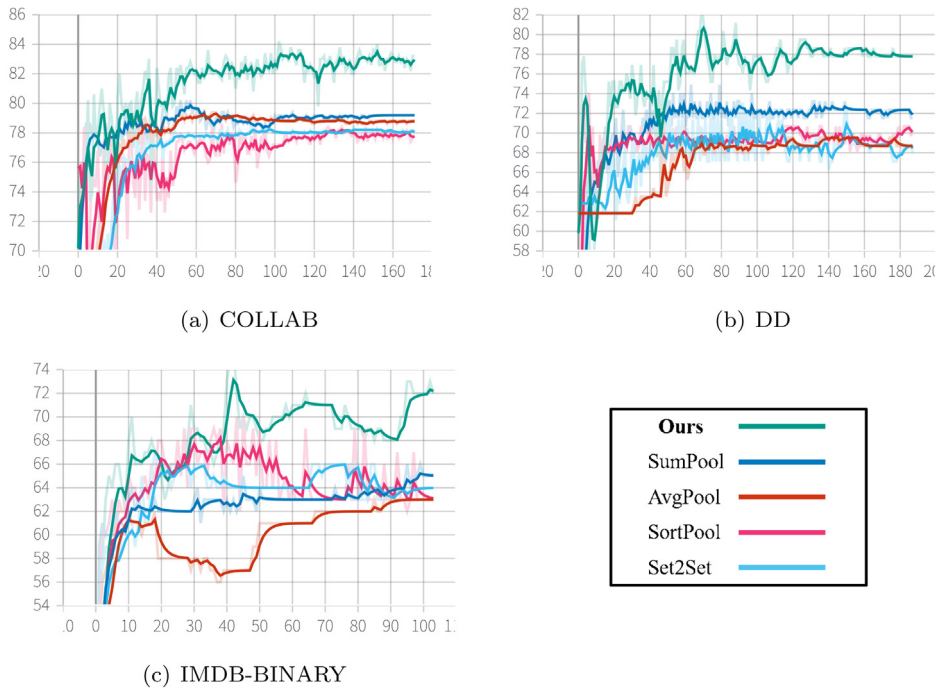

Fig. 1. The performances on testing data, without training for GNN layers. We plot the accuracies(%) on the vertical Y-axis, and epochs on the horizontal X-axis. The GNN layers are initialized with uniform sampling from $[-0.1, 0.1]$, while freezing all their parameters. Concretely, five-layer GCN layers are stacked, and Adam is the optimizer for all approaches and the classifier. More experimental settings can be found in Section 6.1.3.

Table 1
Comparison results on eight graph classification datasets. The *global best results* are highlighted with blue boldface, and the black boldface is employed to label the best accuracy in *family comparison*.

	Method	Dataset (Accuracy % ↑)							
		NCI1	PTC	D&D	ENZY.	COLL.	RED-B.	IMDB-B.	IMDB-M.
Compared Models	WL	**84.26**$_{\pm 1.73}$	57.97$_{\pm 1.73}$	78.29$_{\pm 4.71}$	53.43$_{\pm 1.70}$	77.62$_{\pm 1.87}$	81.20$_{\pm 2.70}$	72.80$_{\pm 6.25}$	50.07$_{\pm 3.43}$
	GK	62.06$_{\pm 3.02}$	59.50	74.78$_{\pm 3.42}$	41.03$_{\pm 1.70}$	64.66$_{\pm 1.93}$	78.00$_{\pm 3.40}$	66.20$_{\pm 7.25}$	43.34$_{\pm 6.23}$
	DGK	80.31	55.65	73.50	53.43	73.09	78.04	66.96	44.55
	GAGCN	83.89	60.59	**80.84**	67.33	80.48	90.40	76.10	52.73
	SASGCN	–	61.51	80.71	–	79.60	91.00	74.00	50.43
	DiffPool	79.51	63.47	80.64	62.53	75.48	90.33	75.48	51.40
	DGCNN	74.44$_{\pm 0.47}$	58.59$_{\pm 2.47}$	79.37±0.94	44.02$_{\pm 6.54}$	74.84$_{\pm 2.70}$	91.00$_{\pm 2.28}$	70.70$_{\pm 6.11}$	48.20$_{\pm 3.94}$
	GIN-0	82.70$_{\pm 1.70}$	64.60$_{\pm 7.00}$	76.67$_{\pm 4.09}$	53.17$_{\pm 6.26}$	80.20$_{\pm 1.90}$	92.40$_{\pm 2.50}$	75.10$_{\pm 5.10}$	52.30$_{\pm 2.80}$
	CapsGNN	78.35$_{\pm 1.55}$	–	75.38$_{\pm 4.17}$	54.67$_{\pm 5.67}$	79.62$_{\pm 0.91}$	–	73.10$_{\pm 4.83}$	50.27$_{\pm 2.65}$
GCN-Based	SumPool	79.83$_{\pm 1.23}$	63.68$_{\pm 6.71}$	77.18$_{\pm 3.08}$	50.50$_{\pm 5.48}$	80.52$_{\pm 1.17}$	89.70$_{\pm 2.19}$	73.80$_{\pm 4.16}$	51.27$_{\pm 3.71}$
	AvgPool	76.90$_{\pm 1.48}$	60.20$_{\pm 8.19}$	75.38$_{\pm 3.99}$	51.83$_{\pm 6.02}$	79.40$_{\pm 1.52}$	89.70$_{\pm 2.55}$	73.60$_{\pm 2.67}$	50.53$_{\pm 4.06}$
	Set2Set	79.08$_{\pm 1.47}$	61.94$_{\pm 10.79}$	74.70$_{\pm 3.73}$	52.33$_{\pm 7.04}$	80.98$_{\pm 1.42}$	**92.55**$_{\pm 1.70}$	74.50$_{\pm 3.28}$	51.13$_{\pm 3.75}$
	SortPool	80.66$_{\pm 1.99}$	64.56$_{\pm 6.75}$	75.04$_{\pm 2.61}$	56.33$_{\pm 6.74}$	76.38$_{\pm 2.00}$	85.70$_{\pm 3.14}$	74.20$_{\pm 3.40}$	50.93$_{\pm 4.14}$
	Ours	**81.75**$_{\pm 1.65}$	**66.31**$_{\pm 7.50}$	**79.32**$_{\pm 2.56}$	**59.67**$_{\pm 4.46}$	**82.24**$_{\pm 0.89}$	91.65$_{\pm 1.71}$	**75.50**$_{\pm 4.90}$	**52.33**$_{\pm 3.93}$
GIN-Based	SumPool	81.51$_{\pm 2.40}$	65.41$_{\pm 6.60}$	78.67$_{\pm 3.31}$	56.33$_{\pm 5.04}$	80.48$_{\pm 1.50}$	88.95$_{\pm 1.79}$	74.20$_{\pm 4.09}$	51.33$_{\pm 4.10}$
	AvgPool	**82.55**$_{\pm 1.91}$	63.96$_{\pm 8.55}$	73.68$_{\pm 4.61}$	54.17$_{\pm 7.30}$	80.88$_{\pm 1.42}$	91.30$_{\pm 2.42}$	75.20$_{\pm 3.05}$	52.73$_{\pm 3.49}$
	Set2Set	80.54$_{\pm 2.36}$	66.33$_{\pm 7.14}$	75.47$_{\pm 3.13}$	60.17$_{\pm 6.40}$	81.62$_{\pm 1.87}$	92.45$_{\pm 2.31}$	74.30$_{\pm 5.92}$	51.80$_{\pm 2.57}$
	SortPool	80.92$_{\pm 2.43}$	67.78$_{\pm 7.07}$	77.78$_{\pm 3.09}$	62.50$_{\pm 4.46}$	81.52$_{\pm 1.63}$	89.50$_{\pm 1.85}$	75.10$_{\pm 2.25}$	52.40$_{\pm 3.79}$
	Ours	82.53$_{\pm 2.43}$	**68.82**$_{\pm 7.35}$	**79.09**$_{\pm 4.61}$	**68.00**$_{\pm 4.33}$	**83.24**$_{\pm 1.83}$	**92.90**$_{\pm 1.71}$	**76.70**$_{\pm 3.93}$	**53.00**$_{\pm 3.70}$

operation in a spatial-based view, such as Graph Attention Network(GAT) [10], Graph Isomorphism Network(GIN) [12], GraphSAGE [11] and etc. Except for the graph convolution, various graph coarsening algorithms, which are designed for hierarchical-structural embeddings, have drawn the attention of researchers very recently [30–32]. However, almost all of them mainly focus on learning effective *local-structural embeddings*, a few pay attention to how to construct structural representation at *graph level* [33,34]. Actually, in order to integrate these local embeddings into graph-level representations, a collapse-style global pooling is adopted widely in the GNN models [12,15,16]. As an adverse side effect of that, local-structural information will be over-compressed. By the way, another commonly used method, which adds a dummy supernode that directly links every original nodes [35], essentially equals global pooling methods because of the aggregation operation of GNNs. Later, DGCNN [20] attempted to address this problem of over-compression by proposing a new graph-level representation generation approach, SortPool [20], where local-structural embeddings are sorted by their structural roles and the TOP-K of them are catenated. Besides, set learning methods were also introduced in graph-level representation learning [36–38] We will give some description and analysis for graph neural networks and their graph-level representation generation approaches in more detail in the next preliminaries section.

3. Preliminaries

Let $G = (\mathcal{V}_G, \mathcal{E}_G)$ be a graph with a set of objects $\mathcal{V}_G = \{v_1, v_2, \ldots, v_n\}$ called nodes (or vertices), and an edge set $\mathcal{E}_G = \{(u, v)|u, v \in \mathcal{V}_G\}$ that describes the relationship among nodes. Besides, let $\mathcal{A}_G \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ be the adjacency matrix of G, and $\mathcal{X}_G \in \mathbb{R}^{|\mathcal{V}|\times C}$ denote a matrix, whose vth row vector represents the attribute of node v. We say $G_S = (\mathcal{V}_{G_S}, \mathcal{E}_{G_S})$ is a local structure(or sub-graph) of G if $\mathcal{V}_{G_S} \subseteq \mathcal{V}_G$ and $\mathcal{E}_{G_S} \subseteq \mathcal{E}_G$. In addition, we let $\boldsymbol{z}(v)$ be the embedding of node v, and $\boldsymbol{z}_G$ be the embedding of the entire graph G.

3.1. The connection between graph neural networks and WL test

Weisfeiler–Lehman Test (WL Test) [39] is a fast approximation algorithm for the well-known NP-hard problem, graph isomorphism judgment. In WL test, subtree whose root is based on vertex v is represented with a color $c(v)$, and $c(v)$ is defined iteratively:

$$c^{t+1}(v) = \text{HASH}(\{c^t(v)\} \cup \{c^t(u)_{u\in\mathcal{N}(v)}\}), \tag{1}$$

where $\mathcal{N}(v)$ is the neighborhood set of v. These different colors are designed to distinguish different subtree structures. Note that WL test follows the form of message passing [40], which is a general GNN framework, GNNs has a very close connection with WL test. In fact, Morris et al. proved that WL is a special case of GNN [41]. If we use AGGREGATE$(\cdot)$ and COMBINE$(\cdot)$ operations replace the HASH$(\cdot)$ function, we will get a general GNN framework defined by Xu et al. [12]:

$$\boldsymbol{z}^{t+1}(v) = \text{COMBINE}(\text{AGGREGATE}(\{\boldsymbol{z}^t(u) : u \in \mathcal{N}(v)\}), \boldsymbol{z}^t(v)) \tag{2}$$

where $\boldsymbol{z}^t(v)$ denotes node embedding of v at the tth layer, and $\boldsymbol{z}^0$ is usually initialized with $\mathcal{X}_G$. In some certain, $\boldsymbol{z}(v)$ can be regarded as *continuous color* $c(v)$ that represents a local-structural information [20,41]. In this paper, we mainly utilized two commonly-used GNNs: The first is Graph Convolutional Networks (GCN) [9], which can be described as

$$\boldsymbol{z}^{t+1}(v) = \sigma\left(W \cdot \text{MEAN}\left\{\boldsymbol{z}^t(u), \forall u \in \mathcal{N}(v) \cup \{v\}\right\}\right) \tag{3}$$

where W is a learnable linear parameter matrix, and $\sigma(\cdot)$ is the non-linear function. Here, MEAN$(\cdot)$ adopts a spectral-based method, *i.e.*, $\text{MEAN}(\boldsymbol{z}^t) = \widetilde{\mathrm{D}}^{-\frac{1}{2}}\widetilde{\mathcal{A}}\widetilde{\mathrm{D}}^{-\frac{1}{2}}\boldsymbol{z}^t$, where $\widetilde{\mathcal{A}} = \mathcal{A} + \mathrm{I}$, and $\widetilde{\mathrm{D}}$ is the diagonal degree matrix of $\widetilde{\mathrm{A}}$. The other GNN we used is Graph Isomorphism Network (GIN) [12]:

$$\boldsymbol{z}^{t+1}(v) = \text{MLP}\left(\left(1 + \epsilon^{t+1}\right) \cdot \boldsymbol{z}^t(v) + \sum_{u\in\mathcal{N}(v)} \boldsymbol{z}^t(u)\right) \tag{4}$$

where MLP$(\cdot)$ denotes a multi-layer perceptron, and ϵ is a trade-off parameter for the anchor node v and its neighborhoods.

3.1.1. Commonly-used graph-level representation generation approaches

In general, the local-structural embeddings should be integrated for each graph when using GNNs to learn their graph-level representation. To further dissect the commonly-used graph-level representation generation approaches, and explore how do they affect the graph-level representation learning in GNN models, we give a brief analysis and formal description for them as follows.

Global pooling. Two types of the most commonly-used global pooling in graph domain are AvgPool and SumPool, respectively. AvgPool can be simply described as $\text{AvgPool}(\boldsymbol{z}) = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \boldsymbol{z}(v)$. The graph-level representation generated by AVgPool can be regarded as the arithmetic mean of local-structural embeddings, which is an important thing to describe the local-structural distribution. However, AvgPool cannot capture other specific properties of distribution, such as variance. Moreover, AvgPool is size-independent, which means that it also cannot capture the graph scale as well. But since SumPool has $\text{SumPool}(\boldsymbol{z}) = |\mathcal{V}| \cdot \text{AvgPool}(\boldsymbol{z})$, compared to AvgPool, it is sensitive for graph scale.

Sort pooling. To obtain more structural details, SortPool rids out of the compression, where the graph-level representation of G is defined as follow:

$$\begin{aligned} \boldsymbol{z}_G &= \biguplus_{v_i\in\mathcal{V}, i=1,2,\ldots\mathrm{K}} \boldsymbol{z}(v_i) \\ \textit{subject to} \quad & z(v_1) \preceq z(v_2) \ldots \preceq z(v_\mathrm{K}) \end{aligned} \tag{5}$$

Here, $\biguplus$ means concatenate operation, and $\preceq$ denotes an order that describes the relative relationship of local structures within each graph. The K is a user-defined parameter to constraint and fix the size of graph-level representations for varying graphs. When $|\mathcal{V}| > \mathrm{K}$, the local-structural embeddings with lower rank will be discarded, on the contrary, if $|\mathcal{V}| < \mathrm{K}$, the rest position will be padded with zeros.

It is easy to prove that SortPool is permutation-invariant [20]. And in the situation where all graphs satisfy $|\mathcal{V}_G| < \mathrm{K}$, we can prove that SortPool is injective. According to the theory proposed by [12], injective mapping is described as the most powerful function to distinguish different multisets. However, unfortunately, due to non-alignment, graph similarity cannot be captured very well, overfitting will be cased more easily.

Set learning. Vinyals et al. proposed a model to learn from a permutation-invariant multiset, called Set2Set [37], which follows a iterative scheme. In the tth iteration, it is defined as

$$\begin{aligned} \boldsymbol{q}_t &= \text{LSTM}(\boldsymbol{q}^*_{t-1}) \\ \alpha_{v,t} &= \frac{\exp(f(\boldsymbol{z}(v), \boldsymbol{q}_t))}{\sum_{u\in\mathcal{V}} \exp(f(\boldsymbol{z}(u), \boldsymbol{q}_t))} \\ \boldsymbol{r}_t &= \sum_{v\in\mathcal{V}} \alpha_{v,t}\boldsymbol{z}(v) \\ \boldsymbol{q}^*_t &= \boldsymbol{q}_t \uplus \boldsymbol{r}_t \end{aligned} \tag{6}$$

Here $\boldsymbol{z}_G = \boldsymbol{q}^*_t$, representing the graph-level embedding of G. Set2set learns a weighted average embedding of $\boldsymbol{z}$ to represent the entire graph G using LSTM [42] and attention mechanism [21]. However, similar to SortPool, Set2Set just considers the relationship between local structures within a single graph. The structural semantics and contributions of the same local structures are not aligned between different graphs.

4. Learning from local-structural distribution

4.1. Local-structural distribution and graph similarity

In graph domain, local structure always is the most important handle to analyze the properties of complex graphs. For instance, R-convolution theory [43] defines the similarity $\mathfrak{S}(G_x, G_y)$ between the graphs as

$$\mathfrak{S}(G_x, G_y) = \sum_{x\in S_x}\sum_{y\in S_y} \mathfrak{s}(x, y), \tag{7}$$

where S_x and S_y separately are the multisets of local structures of G_x and G_y, $\mathfrak{s}(x, y)$ equals 1 if x and y are isomorphic otherwise 0. This theory means that the similarity between two graphs can be measured by the number of isomorphic local structure pairs $|\{(x, y)|x \in S_x, y \in S_y\}|$. In the past decades, researchers have further developed numerous graph similarity and graph kernel algorithms based on this theory [44]. However, R-convolution theory has an obvious weakness, that is, two different graphs

would be considered more similar than the selfsame graphs: a simple case is that if two local-structural sets S_{G_x}, S_{G_y} satisfy S_{G_x} is the subset of S_{G_y}, based on this theory, we will have $\mathfrak{S}(G_x, G_x) \leq \mathfrak{S}(G_x, G_y)$, which is counter-intuitive. To solve this problem, local-structural distribution was employed to weigh the similarity between graphs in some studies [45]. According to Wasserstein metric [46], the distance between two distributions can be defined as :

$$W_p(\boldsymbol{u}, \boldsymbol{v}) = \inf_{\gamma \in \Gamma(\boldsymbol{u}, \boldsymbol{v})} \left(\int_{\mathbb{R}\times\mathbb{R}} d(x, y)^p \mathrm{d}\gamma(x, y) \right)^{1/p}, \tag{8}$$

where $\Gamma(\boldsymbol{u}, \boldsymbol{v})$ is the set of joint distributions γ whose marginals are $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. $d(x, y)$ could be any metric between x and y, in that $(x, y) \sim \gamma$. Eq. (8) describes the similarity of two continuous distributions. For discrete distributions, we can use Dirac delta distribution to approximate, $\boldsymbol{u} \approx \frac{1}{k}\sum \delta_{x_i}$, $\boldsymbol{v} \approx \frac{1}{k}\sum \delta_{y_i}$. Since $W_p(\delta_x, \delta_y) = d(x, y)$ [47], Eq. (8) can be written as,

$$\begin{aligned} W_p(\boldsymbol{u}, \boldsymbol{v}) &\approx W_p\left(\frac{1}{k}\sum_{i=1}^{k}\delta_{x_i}, \frac{1}{k}\sum_{j=1}^{k}\delta_{y_i}\right) \\ &= \min_{\sigma \in P_k} \left(\sum_{i=1}^{k} d\left(x_{\sigma_i}, y_i\right)^p \right)^{1/p} \end{aligned} \tag{9}$$

Here P_n is the set of all permutations of $\{1, 2, 3, \ldots, k\}$. Further, just like R-convolution theory, for two local-structural sampling set $\mathcal{S}_{G_x}, \mathcal{S}_{G_y}$ from graphs G_x and G_y, if we define the metric function $d(x, y)$ as $d(x, y) = 1 - \mathfrak{s}(x, y)$, and take it into Eq. (9), the distance between $\mathcal{S}_{G_x}$ and $\mathcal{S}_{G_y}$ can be written as

$$W_p(\mathcal{S}_{G_x}, \mathcal{S}_{G_y}) = \min_{\sigma \in P_k} \left(\sum_{i=1}^{k} \left(1 - \mathfrak{s}\left(x_{\sigma_i}, y_i\right)\right) \right). \tag{10}$$

The solution of the problem described in Eq. (10) can be transformed into a combinatorial optimization that aims to find the maximum matching between the elements of $\mathcal{S}_{G_x}$ and $\mathcal{S}_{G_y}$. When we say that G_x and G_y are the most similar graphs, *i.e.*, $W_p(\mathcal{S}_{G_x}, \mathcal{S}_{G_y}) = 0$, all elements of $\mathcal{S}_{G_x}$ and $\mathcal{S}_{G_y}$ should be totally matched. This metric overcomes the weak point of the R-convolution theory mentioned above and is also very intuitive. Thus, local-structural distribution seems to be a better tool to analyze the similarity between graph-structured data.

The key to solving the problem described in Eq. (10) is to match/align the local structures. However, deciding if two structures are matched/aligned(*i.e.*, isomorphic) is NP-hard. Fortunately, researchers have proposed some fast approximate algorithms for this problem, *e.g.*, Weisfeiler–Lehman test. From Section 3.1, we knew that GNNs almost have the same power as Weisfeiler–Lehman test in distinguishing structures [12], and GNNs have a more powerful ability in capturing structural similarity [41]. Thus, this paper will leverage GNN technique to learn the structural roles of local structures, so as to match/align them according to their structural semantics.

Through the above analysis using Wasserstein metric, we knew that *aligned local structures is beneficial to capturing the global similarity among global structures.* Thus, this analysis inspires us to design a new strategy to introduce the local-structural distribution information into graph-level representations by aligning the local structures. But we also should note that too accurate statistics of the local-structural distributions will affect the generalization capability of entire models. Thus, this paper adopts an approximate alignment strategy for local structures based on their structural roles to address the overfitting problem, it is named as Batch-Align Pool (BAPool).

4.2. Proposed BAPool

The central idea of BAPool contains two parts: first, discretizing the local-structural embedding of batch graph data based on their structural roles; second, aligning and integrating these discretized local-structural embeddings to get their distribution statistics, so as to encode them into graph embedding representations. In addition to improving the efficiency of statistics, another purpose of the first part is to adjust the statistical granularity to reduce the risk of overfitting. Moreover, the integration operation is utilized to guarantee the uniform size of outputs. Most importantly, the analysis in the previous section shows that the local-structural alignment can make the distance between local-structural distributions become directly quantifiable, which is beneficial to mine the global structural similarity between graphs.

4.2.1. Discretization

Specifically, we expect that the discretization operation in BAPool is equivalent to a cluster function that can map those local structures with similar structural roles into the same cluster. Although a clustering algorithm, like k-means [48], maybe a feasible plan, considering the optimization of neural networks that depends on *mini-batch-based* stochastic gradient descent, there are still several questions: 1. Due to the absence of the order and definition of clusters, the structures in different batches cannot be aligned; 2. Expensive computation cost will be hard to accept, because we hope our approach can follow an end-to-end training fashion. Fortunately, we drew a lesson from SortPool, where local-structural embeddings learned by GNN are regarded as "continuous WL colors", that is, similar local structures will be embedded closer in embedding space than the dissimilar, the same observation can be found in [10,49]. Hence, we attempt to split the embedding space into a series of disjoint areas, each of which corresponds to a bin that collects the local-structural embeddings belonging to the area:

$$\boldsymbol{\mathcal{B}} = \left\{ \mathcal{B}_i : i = 1, 2, \ldots \lambda, \mathbb{R}_{\mathcal{B}} = \bigcup_{i=1}^{\lambda} \mathbb{R}_{\mathcal{B}_i}, \forall i, j, \mathcal{B}_i \cap \mathcal{B}_j = \emptyset \right\} \tag{11}$$

Here, $\mathcal{B}_i$ denotes the ith bin, $\mathbb{R}_{\mathcal{B}}$ is the space spanned by the local-structural embeddings, and $\mathbb{R}_{\mathcal{B}_i}$ denotes the sub-space that $\mathcal{B}_i$ corresponds to. Compared with k-means cluster, the superiority of this strategy is that for the graphs belonging to the different batches, the order and meaning of the bins are fixed relatively, which can help us approximately align those graph-level representations in structural semantics, even for different batches.

Further, we stitch the outputs of all GNN layers to generate multi-hierarchical local-structural embeddings, *i.e.*, $\boldsymbol{h}(v) = \left[\boldsymbol{z}^L(v), \boldsymbol{z}^{L-1}(v), \ldots, \boldsymbol{z}^1(v)\right]$, where L is the number of stacked GNN layers. It is worth mentioning that $\boldsymbol{z}^L(v)$ can be considered as the *most refined continuous WL colors*[20], which means that $\boldsymbol{z}^L(v)$ best reflects the structural role of the local structure around node v. A reasonable explanation for this idea is that the deeper layer has a larger receptive field than the shallow layer, *i.e.*, $\boldsymbol{h}(v)$ inherently contains multi-scale structural information around node v, and $\boldsymbol{z}^L(v)$ reflects the most comprehensive information of $\boldsymbol{h}(v)$ because of its largest receptive field. Following this idea, we can give a naive version of the discretization operation for BAPool, *i.e.*, splitting embedding space of $\boldsymbol{z}^L$ into λ parts evenly,

$$\mathcal{B}^v = \left\lfloor \frac{\lambda \cdot (\boldsymbol{z}^L(v) - \omega)}{\Omega - \omega} \right\rfloor. \tag{12}$$

Here, $\mathcal{B}^v$ denotes the identity of bin that identifies the local structures around v. $\boldsymbol{z}^L(v) = \boldsymbol{h}^1(v)$ represents the output of the

last convolution layer GNN^L, and its channel is set to 1 for simplification, which follows [20]. ω and Ω in Eq. (12) are the lower and upper bound of the GNN^L separately, *i.e.*, $\omega \leq \text{GNN}^L(G, v) \leq \Omega$. Besides, under the control of user-defined parameter λ, the number of bins can be constrained no more than λ.

However, in practice, the bound of the graph convolution layer's output is expensive to calculate due to a large number of parameters and complex forms. However, Chen et al. pointed out that most GNNs mainly focus on learning tree-like local structures [50], which means that they are very limited in structural types. And considering that GNNs usually generate numerous local structural embeddings in the batch-based processing, the local structural types learned by GNNs from different batches will have homogeneity. Therefore, inspired by the characteristic and Batch Normalization technique [51], we further propose a mini-batch strategy to approximate ω and Ω. In that, $\Omega_{\mathfrak{B}} = \text{MAX}(\boldsymbol{z}^L(v) : \boldsymbol{z}^L(v) \in \mathcal{U}_{\mathfrak{B}})$, and $\omega_{\mathfrak{B}} = \text{MIN}(\boldsymbol{z}^L(v) : \boldsymbol{z}^L(v) \in \mathcal{U}_{\mathfrak{B}})$, here $\mathcal{U}_{\mathfrak{B}}$ is the local-structural embeddings set of mini-batch graphs $\mathfrak{B}$. Similar to Batch Normalization, for a given dataset $(\mathbf{G}^{tr}, \mathbf{G}^{inf})$, where $\mathbf{G}^{tr}, \mathbf{G}^{inf}$ respectively represent training data and testing data, the $\omega_{\mathfrak{B}}^{inf}$ and $\Omega_{\mathfrak{B}}^{inf}$ is approximated by the expected values learned from training data $\mathbf{G}^{tr}$ when inferencing,

$$\begin{aligned} \omega_{\mathfrak{B}}^{inf} &= \mathbb{E}\left[\omega_{\mathfrak{B}}^{M,tr}\right] \approx \frac{1}{K}\sum_{i=1}^{K} \omega_{\mathfrak{B}_i}^{M,tr}, \\ \Omega_{\mathfrak{B}}^{inf} &= \mathbb{E}\left[\Omega_{\mathfrak{B}}^{M,tr}\right] \approx \frac{1}{K}\sum_{i=1}^{K} \Omega_{\mathfrak{B}_i}^{M,tr} \end{aligned} \tag{13}$$

Here K is the number of the mini-batches in $\mathbf{G}^{tr}$. Taking $\omega_{\mathfrak{B}}^{inf}$ and $\Omega_{\mathfrak{B}}^{inf}$ into Eq. (12), the discretization results of testing data can be calculated out.

4.2.2. Alignment and integration

After the discretization, each $\boldsymbol{z}(v) \in \mathcal{U}_{\mathfrak{B}}$ will be assigned into a certain local-structural bin $\mathcal{B}_i$ $(1 \leq i \leq \lambda)$. For any $\boldsymbol{h}(v_i), \boldsymbol{h}(v_j) \in \mathcal{U}_{\mathfrak{B}}$ satisfying $v_i \in \mathcal{V}_{G_I}, v_j \in \mathcal{V}_{G_J}, G_I, G_J \in \mathfrak{B}$, we consider that $\boldsymbol{h}(v_i), \boldsymbol{h}(v_j)$ are aligned/matched if $\mathcal{B}^{v_i} = \mathcal{B}^{v_j}$. Just like the spectral histogram statistics, we approximate the local-structure distribution of each graph with the bin-based histogram of local-structural embeddings. Concretely, we integrate the structural information for each bin:

$$\boldsymbol{z}_{\mathcal{B}_i} = \sum_{v \in \mathcal{V}_G, \mathcal{B}^v \in \mathcal{B}_i} \boldsymbol{h}_v \tag{14}$$

Here, we call $\boldsymbol{z}_{\mathcal{B}_i}$ as bin embedding. In order to preserve the distribution properties, we use *sum operation* to collect the embeddings of each bin in Eq. (14). Since 1. we can change the parameter λ to adjust the compressed granularity; 2. the local structures belonging to the same bin have similarities, less information loss will be caused by the compression than directly aggregating operation. Further analysis will be given in Section 5.

Lastly, graph-level representation of G can be generated via stacking those bin embeddings, $\boldsymbol{z}_G = \left[\boldsymbol{z}_{\mathcal{B}_1}; \boldsymbol{z}_{\mathcal{B}_2}; ...; \boldsymbol{z}_{\mathcal{B}_\lambda}\right]$, $\boldsymbol{z}_G \in \mathbb{R}^{\lambda \times d_H}$, where d_H is the cardinality of $\boldsymbol{h}$. Besides, it is worth mentioning that due to the structural-semantic alignment, the importance of different local-structural bins can be directly learned in a supervised manner. For downstream tasks, we can apply learnable parameters $\text{W} \in \mathbb{R}^{c \times d_H}, \text{W} = [\boldsymbol{w}_1; \boldsymbol{w}_2; ...; \boldsymbol{w}_c]$ to fuse $\boldsymbol{z}_G$,

$$\boldsymbol{z}'_G = \biguplus_{i=1}^{\lambda} \sum_{j=1}^{c} \left(\boldsymbol{w}_{j,\cdot} \odot \boldsymbol{z}_G^{i,\cdot}\right) \tag{15}$$

where $\boldsymbol{z}_G^{i,\cdot}$ denotes the ith row of $\boldsymbol{z}_G$ as well as denoting $\boldsymbol{z}_{\mathcal{B}_i}$ of G; $\odot$ is the element-wise product; c is the number of channels, which aims to learn weights from different aspects. In the end, the $\boldsymbol{z}'_G$ will be feed into downstream networks.

4.3. Loss for downstream tasks

In our approach, except for the distribution information, continuous local-structural features are considered as well, which benefits to further capture graph similarity [41]. In order to make full use of these continuous features, we design a loss $\mathcal{L}_d$ for graph classification problem. Specifically, we let $\boldsymbol{z}_{\mathcal{B}_i}^{\mathfrak{B}^{\mathcal{I}}} = \frac{1}{|\mathfrak{B}|}\sum_{G_j^{\mathcal{I}} \in \mathfrak{B}} \boldsymbol{z}_{\mathcal{B}_i}^{G_j^{\mathcal{I}}}$, where $\mathcal{I}$ denotes the category. Here, $\boldsymbol{z}_{\mathcal{B}_i}^{\mathfrak{B}^{\mathcal{I}}}$ is used to represent the anchor of the local-structural embeddings that belong to the ith bin in batch $\mathfrak{B}$. On the one hand, we hope that for the graphs with the same category, their corresponding bin embeddings are closer to the anchors:

$$\mathcal{D}_{in} = \min_{\Theta} \frac{1}{\mathcal{C}} \sum_{\mathcal{I}=1}^{\mathcal{C}} \exp\left(\frac{1}{\lambda N_{\mathcal{I}}} \sum_{i=1}^{N_{\mathcal{I}}} \sum_{j=1}^{\lambda} \left\| \boldsymbol{z}_{\mathcal{B}_j}^{G_i^{\mathcal{I}}} - \boldsymbol{z}_{\mathcal{B}_j}^{\mathfrak{B}^{\mathcal{I}}} \right\|_2\right), \tag{16}$$

where $N_{\mathcal{I}}$ denotes the number of the graphs labeled as $\mathcal{I}$-th class in batch $\mathfrak{B}$. On the other hand, we also hope to increase the gap of different anchors, so that make them focus on learning various local-structural features:

$$\mathcal{D}_{bt} = \max_{\Theta} \frac{1}{\mathcal{C}^2} \sum_{\mathcal{I}=1}^{\mathcal{C}} \sum_{\mathcal{J}=1}^{\mathcal{C}} \exp\left(\frac{1}{\lambda(\lambda-1)} \sum_{i=1}^{\lambda} \sum_{j \neq i}^{\lambda} \left\| \boldsymbol{z}_{\mathcal{B}_i}^{\mathfrak{B}^{\mathcal{I}}} - \boldsymbol{z}_{\mathcal{B}_j}^{\mathfrak{B}^{\mathcal{J}}} \right\|_2\right). \tag{17}$$

We define the loss $\mathcal{L}_d = \log(1 + \mathcal{D}_{in}/\mathcal{D}_{bt})$, and the final loss function for graph classification is $\mathcal{L}_{cls} = \mathcal{L}_c + \alpha\mathcal{L}_d$, where $\mathcal{L}_c$ is the cross entropy loss for training classification networks, and α is trade-off coefficient. Besides, for graph regression task, **M**ean **A**bsolute **E**rror (MAE) is employed as the regression loss, $\mathcal{L}_{reg} = \frac{1}{|\mathbf{D}|}\sum_{i=1}^{n} |\boldsymbol{y}_i - f(G_i)|$, where $f(G_i)$ denotes the prediction given by the models.

5. Analysis of proposed approach

5.1. Permutation invariant analysis

Theorem 1. BAPool *is permutation-invariant when inferencing.*

Proof. Let $\mathbf{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$ be the input of BAPool. For arbitrary permutation pair of $\mathbf{H}$, *e.g.*, $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$, existing a bijection $f : \boldsymbol{\pi} \leftrightarrow \boldsymbol{\pi}'$, having $\boldsymbol{h}_{\pi_i} = \boldsymbol{h}_{f(\pi'_j)}$. Based on Eqs. (12) and (13), we have $\mathcal{B}^{\boldsymbol{h}_{\pi_i}} = \mathcal{B}^{\boldsymbol{h}_{f(\pi'_j)}}$ when inferencing, and we will get the same discretization results for $\mathbf{H}$ under permutation $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$. Since we use permutation-invariant sum operation to integrate embeddings in Eq. (14), we will have $\text{BAPool}(\mathbf{H}|\boldsymbol{\pi}) = \text{BAPool}(\mathbf{H}|\boldsymbol{\pi}')$. Thus, BAPool is permutation-invariant when inferencing.

5.2. Discriminative power analysis

Xu et al. dissected the discriminative power of aggregation operations on learning multiset, and pointed out that *sum operation* is most powerful among them [12]. Here, we will give a brief proof that BAPool has much discriminative power than *sum operation*.

Definition 1 (*Discriminative Power*)**.** Given a graph set $\mathcal{G}$, and a non-isomorphic graph pair set $\mathbf{R} = \left\{(G_i, G_j) : \forall G_i, G_j \in \mathcal{G}, \mathfrak{s}(G_i, G_j) = 0\right\}$, for a mapping $g : \mathbf{H} \to \mathbb{R}$, its discriminative power is defined as

$$\chi(g) = \sum_{(G,G') \in \mathbf{R}} \phi\left(g(f(G_i)), g(f(G_j))\right), \quad \phi(x, y) = \begin{cases} 0 & \text{if x = y} \\ 1 & \text{otherwise} \end{cases}$$

Here, $f(\cdot)$ denotes the local-structural encoder.

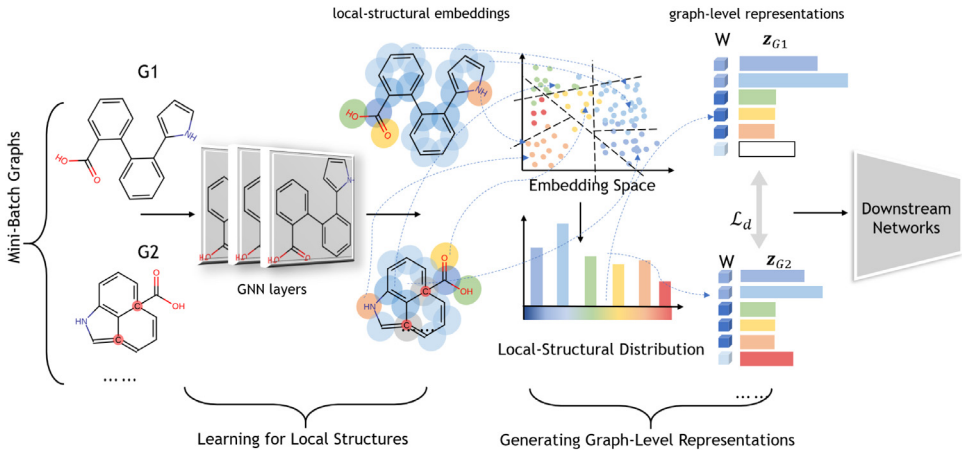

Fig. 2. Overall of proposed framework that learns graph-level representations from local-structural distribution using graph neural networks.

Theorem 2. *The lower bound of discriminative power of BAPool $g_b(\cdot)$ is greater than global SumPool $g_s(\cdot)$,* i.e., $\chi(g_b) \geq \chi(g_s)$.

Proof. Let $\mathbf{S} = \left\{\mathbf{H}_{G_1}, \mathbf{H}_{G_2}, \ldots, \mathbf{H}_{G_n}\right\}$ be a multiset, where the element $\mathbf{H}_{G_i}$ denotes the local-structural embeddings set of graph G_i. Firstly, we prove that $\forall(G, G') \in R$ and $g_s(\mathbf{H}_G) \neq g_s(\mathbf{H}_{G'})$, we have $g_b(\mathbf{H}_G) \neq g_b(\mathbf{H}_{G'})$ as well.

$$g_s(\mathbf{H}_G) \neq g_s(\mathbf{H}_{G'})$$
$$\Rightarrow \sum_{i=1}^{\lambda} \sum_{\boldsymbol{z} \in \mathcal{B}_i^{(G)}} \boldsymbol{z} \neq \sum_{i=1}^{\lambda} \sum_{\boldsymbol{z} \in \mathcal{B}_i^{(G')}} \boldsymbol{z}$$
$$\Rightarrow \boldsymbol{z}_{\mathcal{B}_1}^{(G)} + \boldsymbol{z}_{\mathcal{B}_2}^{(G)} + \cdots + \boldsymbol{z}_{\mathcal{B}_\lambda}^{(G)} \neq \boldsymbol{z}_{\mathcal{B}_2}^{(G')} + \boldsymbol{z}_{\mathcal{B}_2}^{(G')} + \cdots + \boldsymbol{z}_{\mathcal{B}_\lambda}^{(G')}$$

Obviously, there must exist $(\boldsymbol{z}_{\mathcal{B}_i}^{(G)}, \boldsymbol{z}_{\mathcal{B}_j}^{(G)})$ satisfying $(\boldsymbol{z}_{\mathcal{B}_1}^{(G)}) \neq \boldsymbol{z}_{\mathcal{B}_1}^{(G)}$. Thus, we have $\boldsymbol{z}_G \neq \boldsymbol{z}_{G'}$, *i.e.*, $g_b(\mathbf{H}_G) \neq g_b(\mathbf{H}_{G'})$. On the contrary, it is easy to prove that $\exists(G, G')$, $\boldsymbol{z}_G \neq \boldsymbol{z}_{G'}$, but $\sum_{i=1}^{\lambda} \boldsymbol{z}_{\mathcal{B}_i}^{(G)} = \sum_{i=1}^{\lambda} \boldsymbol{z}_{\mathcal{B}_j}^{(G')}$. Thus, we have $\chi(g_b) \geq \chi(g_s)$.

5.3. Complexity analysis

Firstly, since BAPool calculate the bin identity for each local-structural embedding in $\mathcal{O}(1)$ time, the total computational complexity of discretization operation equals $\mathcal{O}(|\mathcal{V}|)$. Secondly, we use sum operation in Eq. (14), and its complexity depends on the number of bins and the dimension of local-structural embeddings, *i.e.*, $\mathcal{O}((\lambda + |\mathcal{V}|)d_H)$. Lastly, the computational complexity of Eq. (15) is $\mathcal{O}(c\lambda d_H)$. In conclusion, the total computational complexity of BAPool equals $\mathcal{O}\left((\lambda(c+1) + |\mathcal{V}|)d_H\right)$.

6. Experiment

In this section, we conduct several empirical studies on both graph classification and regression tasks. All these studies were conducted on a Linux system with an Intel Core i7-8700 CPU, an Nvidia TITAN RTX, and 32 GB RAM.

6.1. Graph classification

Graph classification benchmark is an effective tool to evaluate graph-level representation learning methods. Except for the graph-level representation generation approaches mentioned in Section 3.1.1, we also compare our approach with a series of state-of-the-art (SOTA) graph classification algorithms on a range of graph datasets.

6.1.1. Dataset

We totally employ eight real-world graph classification datasets. There are four real-world datasets in the biological and chemical domain: **NCI-1** collects 4110 chemical compounds that were labeled as active or inactive against non-small cell lung cancer; Two types of compounds in **PTC** are divided according to carcinogenicity about rodents; **D&D** contains 1178 biological macromolecule instances, which are divided into two types, *i.e.*, enzymes versus non-enzymes; **ENZYMES** is a dataset of enzymes that belongs to one of the 6 EC top-level classes. The other four datasets were generated from social networks: **COLLAB** is a scientific-collaboration dataset, where researchers are linked if they have collaborated together, and its task is to classify those scientific collaboration groups as three classes, namely, High Energy Physics, Condensed Mater Physics, and Astro Physics; **IMDB-BINARY** and **IMDB-MULTI** constructs actor relationship networks by connecting actors who appeared together in any movies; Graphs in **REDDIT-BINARY** are generated via online discussions where nodes correspond to users and if anyone responds to another's comment they will be linked.

We use the node attribute $\mathcal{X}_\mathcal{V}$ as the input of models for such datasets in the biological and chemical domain, and for such datasets without node attributes, we use their node degrees as the input

6.1.2. Baseline

The baselines are divided into two types: the first type contains a series of commonly-used graph-level representation generation strategies mentioned in Section 3.1.1, which are employed to compare with our approach under different GNNs; The second type contains following SOTA graph classification algorithms to further evaluate the performance of our approach:

- **WL-subtree Kernel (WL)** [39] is based on the Weisfeiler–Lehman test, which distinguishes non-isomorphic graph via coloring sub-tree structures;
- **Graphlet Kernel (GK)** [45] calculates the graph similarities by preselected graphlet distributions;
- **Deep Graph Kernel (DGK)** [52] is an early work for graph classification using deep learning methods, and this algorithm leverages the dependency between local structures by learning their latent representations;
- **DGCNN** [20] presents an end-to-end framework for graph classification, where SortPool layer was proposed to generate graph-level representations;
- **GAGCN** [34] proposes a graph neural networks based on global-aligned strategy for graph classification task;

- **SASGCN** [33] is a backtrackless aligned-spatial graph convolutional network for graph classification task;
- **DiffPool** [30] implements a differentiable coarseness strategy for GNN via learned assignments based on GraphSAGE;
- **GIN**[12] points out that the sum-aggregator is the best aggregator, and construct graph isomorphism network with sum-aggregator and MLP for graph classification task;
- **CapsGNN** [53] is recent state-of-the-art work for graph classification that introduces the capsule neural network concept into graph domain.

6.1.3. Settings

Settings for Backbone Network. The backbone follows a three-stage framework, as shown in Fig. 2. Besides, we apply two types of commonly-used GNNs to evaluate the performances of the different generation approaches.

In order to compare various approaches under their best settings, we search hyper-parameters from a range. Specifically, we stack {3,4,5} graph neural layers for different datasets and approaches; The hidden channels of GNN layer are selected from {32, 64}; Balance coefficient ϵ in GIN is set to 0; MLP in GIN contains two layers with {32, 64} hidden neurons; The classification network is a two-layer MLP with 128 hidden neurons and LeayReLU(x)($p = 0.1$) is its non-linear function; Before the last layer, dropout technique (the drop ratio equals {0, 0.5}) is also employed to avoid overfitting. α in our classification loss is selected from $\{1e-1, 1e-2\}$. We chose SDG [54] (momentum equals 0.9) or Adam [55] as the optimizer with batch 64 or 256, and the initial learning rate is selected from $\{1e{-}2, 1e{-}3, 1e{-}4\}$.

Settings for Compared Generation Approaches. For Set2Set, we search the number of iterations from $\{2, 3, 4\}$ for different datasets. The hyper-parameter k of SortPool follows the guidelines of original paper, and we set $K = \text{MIN}(500, 0.9 \times \text{MAX}(\{|\mathcal{V}_G| : G \in \mathbf{G}\}))$ for the unmentioned datasets in the guidelines. For BAPool, we set $c = 32$, and select λ from $\{m, m + 10, m + 20\}$, where $m = \text{MEAN}(\{|\mathcal{V}_G| : G \in \mathbf{G}\})$. Further analysis about λ is shown in Section 6.4.

Settings for Compared Models. For graphlet kernel, we calculate the graphlet distribution via searching the graphlets with three nodes for each graph; The iterations of WL-subtree Kernel is searched from $\{1, 2, 3, 4, 5\}$. $\mathcal{C}$-SVM [56] is employed as the classifier for such graph kernel methods, where $\mathcal{C}$ is selected from $\{1e-3, 1e-2, \ldots, 1e3\}$. We report the results corresponding to the best parameter settings. For GNN-based models, if their official source codes are available, we re-run these codes under the same evaluating standards as ours, and their hyper-parameters follow the guidelines proposed in their papers or codes. *In order to get a more convincing conclusion, we reported the* ***best*** *accuracy between the results we got and the results reported in their papers.* If we cannot find the official implements of some compared algorithms, we report the published results directly.

All approaches are evaluated under 10-fold cross-validation, and we report the average result and standard deviation over 10 folds.

6.2. Result analysis

The experimental results are reported in Table 1, from which we have the observations as follows: first of all, the choice of both GNN and graph-level representation generation approaches affect the performance in this task. Besides, GNN-based approaches show more competitive performance than graph kernels no matter in the chemical domain and social networks. Compared to existing graph-level representation generation approaches, our proposed BAPool consistently achieved the best performance among most datasets under both GCN and GIN settings. Furthermore, although both SortPool and our approach use 2D-tensor as the graph-level representations, the most difference between them is the structural-semantic alignment. To further demonstrate this point, we visualize the local-structural embeddings (*i.e.*, $\boldsymbol{h}$). In that, we marked the local-structural embeddings as the same color if they are thought to have the same or similar structural role. As shown in Fig. 3, in our approach, the local-structural embeddings with the same color are clustered together broadly, whereas they are messy in SortPool, which illustrates that for different graph data, their graph-level representations generated by our approach are more aligned in structural-semantic than SortPool. According to Wasserstein metric, our approach can capture graph similarity better. Thus, our approach significantly outperforms SortPool on most datasets. Besides, compared to Set2Set that calculates weights using Attention and LSTM, our approach can learn weights for different local structures in a supervised way because of aligned structural-semantic. Due to getting rid of complex computations for Attention and LSTM, our approach runs faster than Set2Set, and gets higher accuracies on almost all datasets than Set2Set.

Table 2
Regression results for graph size with the GIN-backbone.

Dataset (MAE ↓)				
Statistic	IMDB-BINARY	REDDIT-BINARY	D&D	ENZYMES
#Avg. Size	212.8	1425.1	1715.6	156.9
#Std. Size	218.9	1798.7	1657.6	65.0
SumPool	6.31E−2	2.40E−2	1.00E−2	5.76E−1
AvgPool	1.14E−1	7.40E−2	4.28E−1	6.99E−1
SortPool	1.00E−1	7.77E−2	2.04E−1	5.20E−1
Set2Set	6.46E−2	1.24E−1	3.97E−1	6.07E−1
ours	**4.58E−2**	**5.76E−3**	**9.53E−3**	**1.35E−1**

6.3. Graph regression

We conduct two regression experiments, including an artificial task to evaluate the perception of graph scale for different approaches, and real-world graph regression task in the chemical domain.

6.3.1. Regression for graph scale

This task can be expressed as follows: for a given graph G, the model f should answer the scale of G, where the scale of G is defined as $|\mathcal{V}_G| + |\mathcal{E}_G|$, including both the node and edge quantities of G. In this experiment, in order to accelerate the convergence speed, graph scales of graph data are scaled by z-score standardization. The configurations of the model follow the description of Section 6.1.3 as well.

There are three purposes of this experiment: firstly, we hope empirically prove that sum operation can capture the size of the input graph, which is key for BAPool to construct local-structural distribution information; secondly, the graph scale as the most-intuitive global-structural information of graph data can shed light on evaluating the ability of different approaches to learning global-structural features; Lastly, we hope to demonstrate the importance of the choice of the graph-level representation generation approach.

From the experimental results shown in Table 2, we can see that our proposed BAPool achieved the first good performance on all four datasets. In addition, SumPool has much less MAE loss than other compared approaches, which means that SumPool is much more scale-sensitive, and this accords with our previous analysis. But interestingly, Set2Set seems not to be able to capture this global property very well. The reason may be that it tends

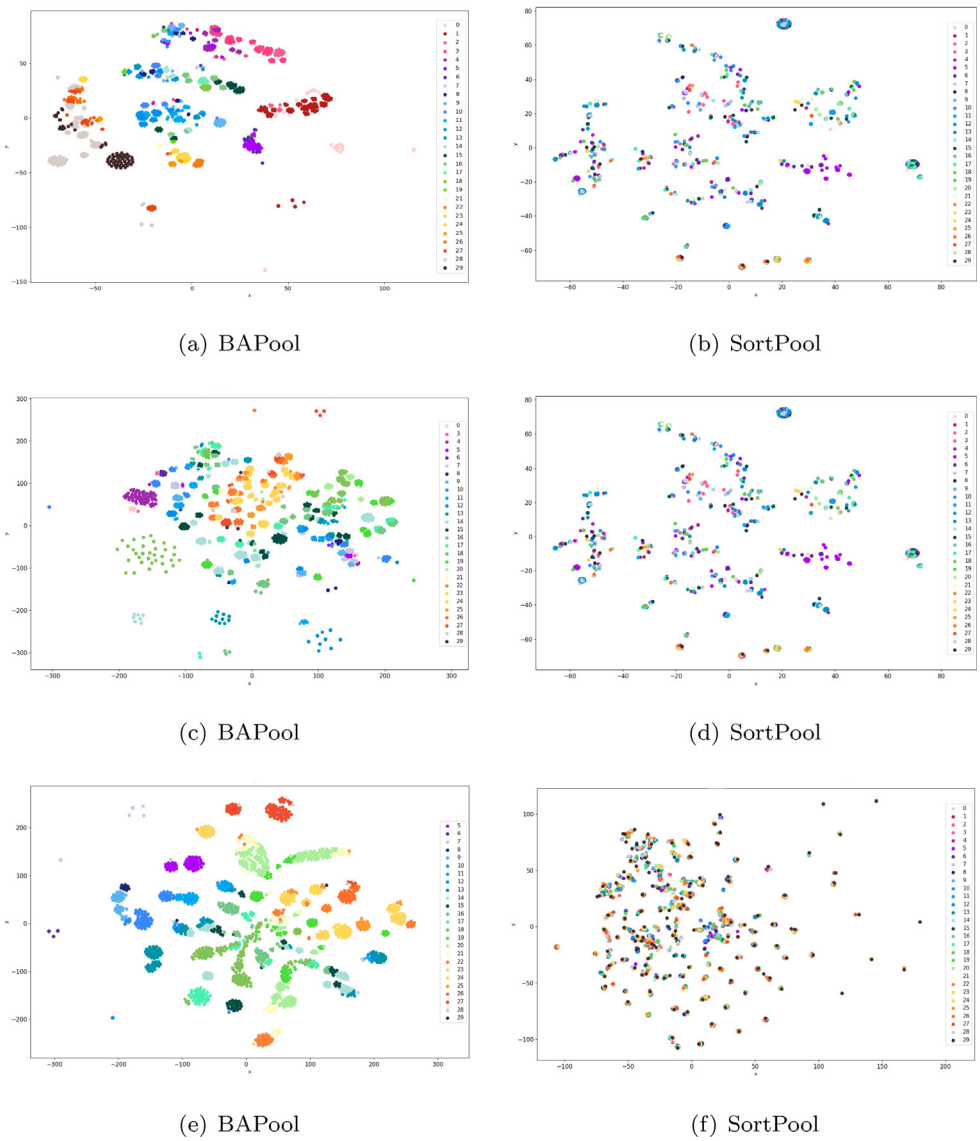

Fig. 3. T-SNE visualization of node embeddings. Here, each dot corresponds to a local-structural embedding. The colors denote the structural roles of those embeddings, which are calculated by BAPool ($\lambda = 30$) or SortPool ($K = 30$). These visualizations are generated using scikit-learn [57], and we follow the default settings, and the sub-figures (a)(b), (c)(d), (e)(f) respectively correspond to IMDB-BINARY, IMDB-MULTI, and COLLAB datasets.

to focus on local-structural details, just like SortPool. From these observations, we have the reasons to believe that different graph-level representation generation strategies make models focus on different graph properties — this also underlines the need for researching them.

6.3.2. Regression on ZINC dataset

ZINC is a real-world molecular graph dataset that collects 12K commercially available chemical compounds. In this task, models are required to predict the octanol–water partition coefficient (logP) of these compounds, which is usually difficult and expensive to measure in a chemical way. Interestingly, this property usually has a close connection with the chemical characteristic of atoms and fragments(local structures) [58]. Hence, we suppose that a good prediction method on ZINC should learn the local-structural characteristics. This task helps us further explore the ability of BAPool in capturing local-structural information.

In this study, except for four compared graph-level generation approaches (their settings follow Section 6.1.3), we also compare our approach with the three latest state-of-the-art GNNs [59–61] that are designed for this task. The results in their original papers are reported directly because of the standard data split of ZINC. Besides, in order to make full use of the structural information provided by ZINC, we extend GIN to learn from both node and edge attribute:

$$\mathbf{Z}^{t+1}(v) = \mathrm{MLP}\left(\left(1+\epsilon^{(t)}\right)\cdot \mathbf{Z}^{t}(v) + \sum_{u\in\mathcal{N}(v)} \left(\mathbf{Z}^{t}(u) \uplus \mathcal{X}_{\mathcal{E}}(v,u)\right)\boldsymbol{\Theta}\right) \tag{18}$$

where $\mathcal{X}_{\mathcal{E}}(v,u)$ denotes the attribute of edge between v and u, and $\boldsymbol{\Theta}$ denotes a learnable parameter matrix.

The regression results on ZINC dataset are reported in Table 3. In general, the experimental results show that the graph-level representation generation strategies have a significant impact on this real-world task, and with the blessing of BAPool, GIN achieves or even surpasses the prediction performance of the SOTA graph regression models. Furthermore, interestingly, from the experimental result of SortPool, we find that although SortPool as a non-aggregate strategy does not compress the local-structural embeddings, its MAE loss is the largest when GIN does

Table 3
Regression results on ZINC dataset with the GIN-backbone. The "w/o → w/ E.F." means "From w/o Eedge Feature to w/ Eedge Feature".

Method	ZINC (MAE ↓)					
	w/o edge feature		w/ edge feature		w/o → w/ E.F. Improvement Ratio (%)	
	VAL	TEST	VAL	TEST		
GatedGCN	–	4.22E−1	–	3.63E−1		
PNA	–	3.20E−1	–	1.88E−1		
DGN	–	2.19E−1	–	**1.68E−1**	VAL	TEST
SumPool	2.94E−1	2.98E−1	2.48E−1	2.35E−1	15.6	21.1
AvgPool	2.98E−1	2.72E−1	2.66E−1	2.53E−1	10.7	6.9
SortPool	3.35E−1	3.23E−1	2.26E−1	2.23E−1	32.5	30.9
Set2Set	2.90E−1	2.60E−1	2.10E−1	2.17E−1	27.5	16.5
Ours	**2.51E−1**	**2.16E−1**	**1.78E−1**	1.70E−1	29.0	21.3

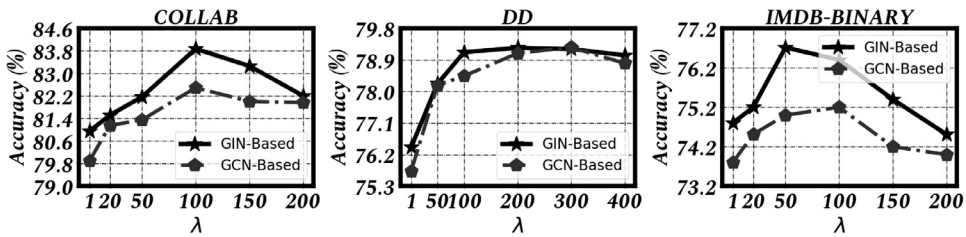

Fig. 4. λ Impact.

not consider the edge attributes. A reasonable explanation for this phenomenon is that GIN cannot sufficiently learn the local local-structural information of graphs due to missing the attributes for edges, and SortPool also cannot express global structural information well because of the lack of natural alignment, so it is easier to overfit. However, the performance of SortPool will be significantly improved when GIN considers the edge attributes, which further illustrates the importance of local-structural information for SortPool. Then, through analyzing the improvement ratio, we can observe that with the enhance for local-structural information mining, the BAPool and SortPool can obtain more significant improvements than the simple aggregation strategies such as SumPool and AvgPool, which confirms our previous point that these strategies have stronger ability of these strategies in expressing the local structures.

6.4. Parameter impact study for λ

The hyper-parameters λ control the statistical granularity of the local-structural distribution. To explore how does λ affect the performance, and give a guideline for the λ setting, we conduct an experiment to study it. We sample λ from a range according to the graph scale of datasets and froze all other hyper-parameters. The line chart of the performance on several graph datasets is drawn as Fig. 4. As we can see, it is clear that the fluctuation of accuracy is unimodal over the large range of λ and the accuracy goes to peak at the middle and decreased with the increment of λ subsequently. This is because that the BAPool will degenerate as SumPool when λ trends to 1, which leads to over-compression of structural information, however, BAPool also could be slightly overfitting when λ become too large due to too many structural details. More specifically, we can observe that BAPool almost achieves the best performance when λ equals around a median value, particularly close to the average of graph node scale.

7. Conclusion and future work

Considering the drawbacks of the existing graph-level representation generation approaches, *i.e.*, over-compression of local-structural information and the lack of natural alignment between graphs, this paper proposed a novel approach that learns structural-aligned representation for each whole graph by leveraging the local-structural distributions. On the one hand, benefiting from the non-collapse-type operations, more local-structural information are preserved. On the other hand, the distribution information encoded in the graph-level representations is also beneficial to capturing global graph similarity for the downstream networks. Furthermore, we briefly dissected existing relevant approaches, discussing and comparing their performance on a range of real-world and artificial graph tasks. The experimental results show that our approach broadly outperforms compared methods, and the visualization also backed up our views and analysis. In the end, we conduct hyper-parameter study to further analyze our approach.

In the future, we are going to extend the local-structural richness because general GNNs are experts in digging out tree-like local structures, but they are not capable of capturing other simple graph-theoretic properties, *e.g.*, cycle or cliques [50], which usually are an important for social network analysis [62]. The increase in the types of local structures will enrich the structural information and will significantly improve the expressive power of our approach in graph-level representation learning.

CRediT authorship contribution statement

Wei-Xiang Sun: Conceptualization, Methodology, Coding, Data curation, Writing. **Hui Xue:** Validation, Manuscript revision, Investigation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No. 62076062), Fundamental Research Funds for the Central Universities (Grant No. 2242021k30056), and National Key R&D Program of China (Grant No. 2017YFB1002801). Furthermore, the work was also supported by Collaborative Innovation Center of Wireless Communications Technology.

References

[1] L. Lambers, D. Strüber, G. Taentzer, K. Born, J. Huebert, Multi-granular conflict and dependency analysis in software engineering based on graph transformation, in: Proceedings of the 40th International Conference on Software Engineering, 2018, pp. 716–727.

[2] C.W. Coley, W. Jin, L. Rogers, T.F. Jamison, T.S. Jaakkola, W.H. Green, R. Barzilay, K.F. Jensen, A graph-convolutional neural network model for the prediction of chemical reactivity, Chem. Sci. 10 (2) (2019) 370–377.

[3] X. Yue, Z. Wang, J. Huang, S. Parthasarathy, S. Moosavinasab, Y. Huang, S.M. Lin, W. Zhang, P. Zhang, H. Sun, Graph embedding on biomedical networks: methods, applications and evaluations, BMC Bioinform. 36 (4) (2020) 1241–1251.

[4] A. Breuer, R. Eilat, U. Weinsberg, Friend or faux: graph-based early detection of fake accounts on social networks, in: International World Wide Web Conferences, 2020, pp. 1287–1297.

[5] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.

[6] A. Grover, J. Leskovec, Node2Vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.

[7] L.F. Ribeiro, P.H. Saverese, D.R. Figueiredo, Struc2Vec: learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 385–394.

[8] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3837–3845.

[9] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, 2017.

[10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, 2018.

[11] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.

[12] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in: Proceedings of the 7th International Conference on Learning Representations, 2019.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24.

[14] K.M. Borgwardt, C.S. Ong, S. Schönauer, S.V.N. Vishwanathan, A.J. Smola, H. Kriegel, Protein function prediction via graph kernels, in: Proceedings of the 13th International Conference on Intelligent Systems for Molecular Biology, 2005, pp. 47–56.

[15] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: Proceedings of the 33nd International Conference on Machine Learning, 2016, pp. 2014–2023.

[16] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in: Advances in Neural Information Processing Systems, 2015, pp. 2224–2232.

[17] B. Zhang, Q. Zhao, W. Feng, S. Lyu, Alphamex: a smarter global pooling method for convolutional neural networks, Neurocomputing 321 (2018) 36–48.

[18] Z. Gao, J. Xie, Q. Wang, P. Li, Global second-order pooling convolutional networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3024–3033.

[19] B. Donon, B. Donnot, I. Guyon, A. Marot, Graph neural solver for power systems, in: International Joint Conference on Neural Networks, 2019, pp. 1–8.

[20] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An end-to-end deep learning architecture for graph classification, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 4438–4445.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[22] J. Huang, Z. Li, N. Li, S. Liu, G. Li, AttPool: towards hierarchical feature representation in graph convolutional networks via attention mechanism, in: IEEE/CVF International Conference on Computer Vision, 2019, pp. 6479–6488.

[23] H.L. Morgan, The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service., J. Chem. Doc. 5 (2) (1965) 107–113.

[24] R.C. Glen, A. Bender, C.H. Arnby, L. Carlsson, S. Boyer, J. Smith, Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to ADME, IDrugs 9 (3) (2006) 199–204.

[25] D. Rogers, M. Hahn, Extended-connectivity fingerprints, J. Chem. Inf. Model. 50 (5) (2010) 742–754.

[26] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: Proceedings of the 33nd International Conference on Machine Learning, 2016, pp. 2014–2023.

[27] A.J.-P. Tixier, G. Nikolentzos, P. Meladianos, M. Vazirgiannis, Graph classification with 2d convolutional neural networks, in: International Conference on Artificial Neural Networks, 2019, pp. 578–593.

[28] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: Proceedings of the 2nd International Conference on Learning Representations, 2014.

[29] J. Chen, T. Ma, C. Xiao, FastGCN: fast learning with graph convolutional networks via importance Sampling, in: Proceedings of the 6th International Conference on Learning Representations, 2018.

[30] Z. Ying, J. You, C. Morris, X. Ren, W.L. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in: Advances in Neural Information Processing Systems, 2018, pp. 4805–4815.

[31] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in: Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 3734–3743.

[32] H. Gao, S. Ji, Graph U-nets, in: Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 2083–2092.

[33] L. Bai, L. Cui, Y. Jiao, L. Rossi, E. Hancock, Learning backtrackless aligned-spatial graph convolutional networks for graph classification, IEEE Trans. Pattern Anal. Mach. Intell. (2020).

[34] X. Hui, S. Wei-Xiang, Globally aligned graph convolutional network for graph classification, J. Hunan Univ. Nat. Sci. 06 (2021).

[35] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80.

[36] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, A.J. Smola, Deep sets, in: Advances in Neural Information Processing Systems, 2017, pp. 3391–3401.

[37] O. Vinyals, S. Bengio, M. Kudlur, Order matters: sequence to sequence for sets, in: Proceedings of the 4th International Conference on Learning Representations, 2016.

[38] N. Navarin, D.V. Tran, A. Sperduti, Universal readout for graph convolutional neural networks, in: International Joint Conference on Neural Networks, 2019, pp. 1–7.

[39] B. Weisfeiler, A.A. Lehman, A reduction of a graph to a canonical form and an algebra arising during this reduction, Nauchno-Technicheskaya Inf. 2 (9) (1968) 12–16.

[40] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1263–1272.

[41] C. Morris, M. Ritzert, M. Fey, W.L. Hamilton, J.E. Lenssen, G. Rattan, M. Grohe, Weisfeiler and Leman go neural: higher-order graph neural networks, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 4602–4609.

[42] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[43] D. Haussler, Convolution Kernels on Discrete Structures, Technical Report UCSC-CRL-99-10, Computer Science Department, UC Santa Cruz, Santa Cruz, CA, USA, 1999.

[44] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt, Graph kernels, J. Mach. Learn. Res. 11 (2010) 1201–1242.

[45] N. Shervashidze, S.V.N. Vishwanathan, T. Petri, K. Mehlhorn, K.M. Borgwardt, Efficient graphlet kernels for large graph comparison, in: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, 2009, pp. 488–495.

[46] S. Vallender, Calculation of the Wasserstein distance between probability distributions on the line, Theory Probab. Appl. 18 (4) (1974) 784–786.

[47] R. Mardare, P. Panangaden, G.D. Plotkin, Free complete Wasserstein algebras, Log. Methods Comput. Sci. 14 (3) (2018) 16.

[48] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, no. 14, 1967, pp. 281–297.

[49] Y. Huang, J. Zhang, Y. Yang, Z. Gong, Z. Hao, GNNVis: visualize large-scale data by learning a graph neural network representation, in: The 29th ACM International Conference on Information and Knowledge Management, 2020, pp. 545–554.

[50] Z. Chen, L. Chen, S. Villar, J. Bruna, Can graph neural networks count substructures? in: Advances in Neural Information Processing Systems, 2020, pp. 10383–10395.

[51] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 448–456.

[52] P. Yanardag, S.V.N. Vishwanathan, Deep graph kernels, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1365–1374.

[53] X. Zhang, L. Chen, Capsule graph neural network, in: Proceedings of the 7th International Conference on Learning Representations, 2019.

[54] N. Qian, On the momentum term in gradient descent learning algorithms, Neural Netw. 12 (1) (1999) 145–151.
[55] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, 2015.
[56] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[58] W.M. Meylan, P.H. Howard, Atom/fragment contribution method for estimating octanol–water partition coefficients, J. Pharm. Sci. 84 (1) (1995) 83–92.
[59] X. Bresson, T. Laurent, Residual gated graph convnets, 2017.
[60] G. Corso, L. Cavalleri, D. Beaini, P. Liò, P. Velickovic, Principal neighbourhood aggregation for graph nets, in: Advances in Neural Information Processing Systems, 2020, pp. 13260–13271.
[61] D. Beaini, S. Passaro, V. Létourneau, W.L. Hamilton, G. Corso, P. Liò, Directional graph networks, 2020.
[62] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: simple building blocks of complex networks, Science 298 (5594) (2002) 824–827.