DHAKR: Learning Deep Hierarchical Attention-Based Kernelized Representations for Graph Classification

Feifei Qian[1], Lu Bai[1*], Lixin Cui[2], Ming Li[3,4*], Ziyu Lyu[5], Hangyuan Du[6], Edwin Hancock[7]

[1] School of Artificial Intelligence, and Engineering Research Center of Intelligent Technology and Educational Application, Ministry of Education, Beijing Normal University, Beijing, China;
[2] School of Information, Central University of Finance and Economics, Beijing, China;
[3] Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, China;
[4] Zhejiang Institute of Optoelectronics, Jinhua, China;
[5] School of Cyber Science and Technology, Sun Yat-Sen University, Shenzhen, China;
[6] School of Computer and Information Technology, Shanxi University, Taiyuan, China;
[7] Department of Computer Science, University of York, York, United Kingdom.
feifei_qian@mail.bnu.edu.cn, bailu@bnu.edu.cn, cuilixin@cufe.edu.cn, mingli@zjnu.edu.cn, lvzy7@mail.sysu.edu.cn, duhangyuan@sxu.edu.cn, edwin.hancock@york.ac.uk

Abstract

Graph-based representations are powerful tools for analyzing structured data. In this paper, we propose a novel model to learn Deep Hierarchical Attention-Based Kernelized Representations (DHAKR) for graph classification. To this end, we commence by learning an assignment matrix to hierarchically map the substructure invariants into a set of composite invariants, resulting in hierarchical kernelized representations for graphs. Moreover, we introduce the feature-channel attention mechanism to capture the interdependencies between different substructure invariants that will be converged into the composite invariants, addressing the shortcoming of discarding the importance of different substructures arising in most existing R-convolution graph kernels. We show that the proposed DHAKR model can adaptively compute the kernel-based similarity between graphs, identifying the common structural patterns over all graphs. Experiments demonstrate the effectiveness of the proposed DHAKR model.

Introduction

Graph-based representations have been widely employed in various research domains, such as social networks (Maleki, Padmanabhan, and Dutta 2022), molecular chemistry (An et al. 2024; Kosmala et al. 2023; Kelvinius et al. 2023), recommendation systems (Wu et al. 2019; Wei et al. 2022), etc. One way to effectively capture the structural characteristics for graph data analysis is to employ graph kernels.

Graph kernels aim to measure the similarity between graphs by mapping their structural information into a high-dimensional Hilbert space. Under this scenario, most existing graph kernels are based on counting the pairs of isomorphic substructures decomposed from original graphs. This is the so-called R-convolution framework proposed by (Haussler et al. 1999). Specifically, given two sample graphs G_m and G_n, a R-convolution graph kernel K_{R} is defined as

$$K_{\mathrm{R}}(G_m, G_n) = \sum_{g_m \subseteq G_m} \sum_{g_n \subseteq G_n} s(g_m, g_n), \qquad (1)$$

where $s(g_m, g_n)$ denotes the similarity between the substructures g_m and g_n. Based on different types of substructures, most R-convolution kernels can be divided as kernels based on walks, paths, and subgraphs or subtrees. For instance, (Gärtner, Flach, and Wrobel 2003) have proposed a Random Walk Graph Kernel (RWGK) that compares the sequences of visited nodes during random walks. (Shervashidze et al. 2009) have proposed a Graphlet Count Graph Kernel (GCGK) by counting the frequency of small connected subgraphs (i.e., graphlets). (Borgwardt and Kriegel 2005) have developed the Shortest Path Graph Kernel (SPGK) (Borgwardt and Kriegel 2005) by counting the pairs of common shortest paths. (Shervashidze et al. 2011) have defined the Weisfeiler-Lehman Subtree Kernel (WLSK) (Shervashidze et al. 2011) by iteratively counting the number of pairwise isomorphic subtrees extracted from the Weisfeiler-Lehman (WL) algorithm. Other alternative R-convolution kernels also include the Optimal Assignment Kernel (OAK) (Kriege, Giscard, and Wilson 2016), the Subgraph Alignment Kernel (SAK) (Kriege and Mutzel 2012), the Wasserstein WLSK (Togninalli et al. 2019), etc.

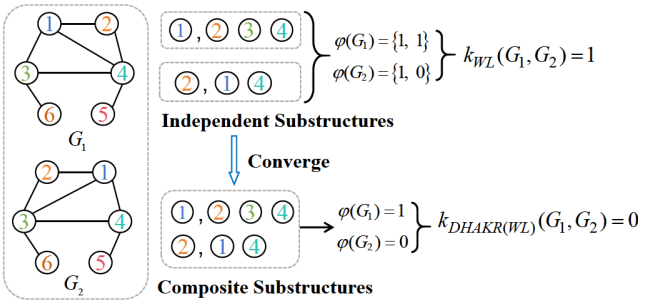


Figure 1: Illustrative example of the motivation.

*Corresponding authors: Lu Bai, Ming Li

However, there are some common drawbacks arising in existing R-convolution kernels. First, these kernels only focus on counting the number of pairwise isomorphic substructures, neglecting the global structure information. As shown in Fig.1, there are two non-isomorphic graphs, G_1 and G_2. We assume $\{1, 234\}$ as substructure s, $\{2, 14\}$ as substructure s'. G_1 and G_2 have the isomorphic substructure s, contributing one unit to the WLSK kernel value, even though the global structures are not isomorphic. Second, these kernels cannot identify the significance of different substructures, neglecting the influence of redundant features. Third, the kernel construction is separated from the classifier (e.g., C-SVMs). Although graph deep learning methods can provide an end-to-end learning framework, they usually sum up the node features to derive graph representations, resulting in topological information loss.

The aim of this paper is to address the above drawbacks, by proposing a novel Deep Hierarchical Attention-based Kernelized Representation (DHAKR) model. One key innovation is the hierarchical framework that can converge the homogeneous substructures into new composite structures. For instance, as shown in Fig.1, the two independent substructures will be combined as a composite substructure $\{\{1, 234\}, \{2, 14\}\}$. Since G_2 does not contain the composite substructure, then the kernel value is 0. This indicates that the composite features are effective for identifying non-isomorphic global structures. As a result, the proposed model can simultaneously capture the global information at higher levels and the local information at lower levels. Specifically, the main contributions of this paper are threefold.

First, we define an assignment matrix that can converge homogeneous substructures into a set of composite substructures. Moreover, to identify the importance of different substructures, we employ the attention mechanism to assign different weights to the homogeneous substructures. Since the substructures are converged associated with different weights, the resulting composite substructure can significantly discriminate the importance of the original substructures. By hierarchically performing the above computational procedure, we can compute a family of hierarchical attention-based substructure invariants for original graphs. **Second**, with the above hierarchical substructure invariants to hand, we define a novel kernel-based learning framework to compute the DHARK for graphs, by computing the R-convolution kernel between pairwise graphs associated with the hierarchical invariants. We show that the resulting hierarchical kernel matrixes can be seen as the kernel-based similarity embedding vectors of all sample graphs, and are differentiable. As a result, the proposed DHAKR model is end-to-end trainable. **Third**, we demonstrate that the DHAKR can outperform state-of-the-art graph classification methods.

Literature Reviews of Related Works

The R-convolution Graph Kernels

We briefly review two classical R-convolution kernels that are closely related to this work, including the WLSK and SPGK kernels. The WLSK kernel employs the iterative label refinement and measures the similarity between subtree patterns. Assuming $l_0(u)$ denotes the initial label of node u, for each iteration t, the WLSK concatenates the node's current label with the sorted multiset of its neighbors' labels $\mathcal{L}_{\mathcal{N}}^{t-1}(u))$. Then the WLSK maps each unique concatenated label into a new label through a hash function, i.e.,

$$l_t(u) = \mathrm{Hash}(l_{t-1}(u), \mathcal{L}_{\mathcal{N}}^{t-1}(u)), \tag{2}$$

where $l_t(u)$ refers a subtree rooted at u of height t. Given a pair of graphs G_m and G_n, the WLSK kernel is computed by counting the number of common subtree patterns, i.e.,

$$K_{\mathrm{WL}} = \sum_{t=0}^{T_{max}} \sum_{i=0}^{|\mathcal{L}^t|} c(G_m, l_t^i) c(G_n, l_t^i), \tag{3}$$

where T_{max} represents the maximum iteration number, l_t^i denotes the i-th node label of the iteration t, and $c(G_m, l_t^i)$ counts the subtree pattern labelled by l_t^i in G_m.

The idea of the SPGK kernel is to compare the isomorphic shortest paths within pairwise graphs. Given a pair of graphs G_m and G_n, the SPGK kernel is defined as

$$K_{\mathrm{SP}} = \sum_{p_i \in \mathcal{P}} c(G_m, p_i) c(G_n, p_i), \tag{4}$$

where $\mathcal{P}$ denotes the set of all possible shortest paths appearing in all graphs, $c(G_m, p_i)$ is the function that counts the number of the shortest path of length p_i in G_m.

Remarks: Both the WLSK and SPGK kernels tend to use all substructures to compute the kernel matrix, without considering the varying roles of different substructures. Furthermore, the construction of the kernel matrix and the classifier are independent from each other, these kernels cannot provide an end-to-end kernelized learning architecture.

The Graph Neural Networks (GNNs)

GNNs have been widely employed for handling graph structures (Niepert, Ahmed, and Kutzkov 2016). Specifically, they leverage either spectral or spatial-based convolution strategies to learn effective graph representations and have achieved outstanding performance for graph classification. For instance, (Atwood and Towsley 2016) have proposed the Diffusion Convolution Neural Network (DCNN) that utilizes the diffusion convolution operation to propagate the information across the graph. (Zhang et al. 2018) have proposed the Deep Graph Convolution Neural Network (DGCNN) that employs a unique SortPooling layer to sort graph nodes into a consistent order based on the substructure information. (Xu et al. 2019) have proposed the Graph Isomorphism Network (GIN) to maximize the ability to distinguish different graph structures, making it as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test.

Remarks: Although the above GNNs can provide an end-to-end learning framework, they still suffer from other drawbacks. First, the GIN and DCNN tend to directly sum up the node features as the global representation, not only discarding local node information but also ignoring the structural differences of different nodes. Second, the DGCNN only preserves the local structure information residing on the top-ranked nodes, resulting in topological information loss.

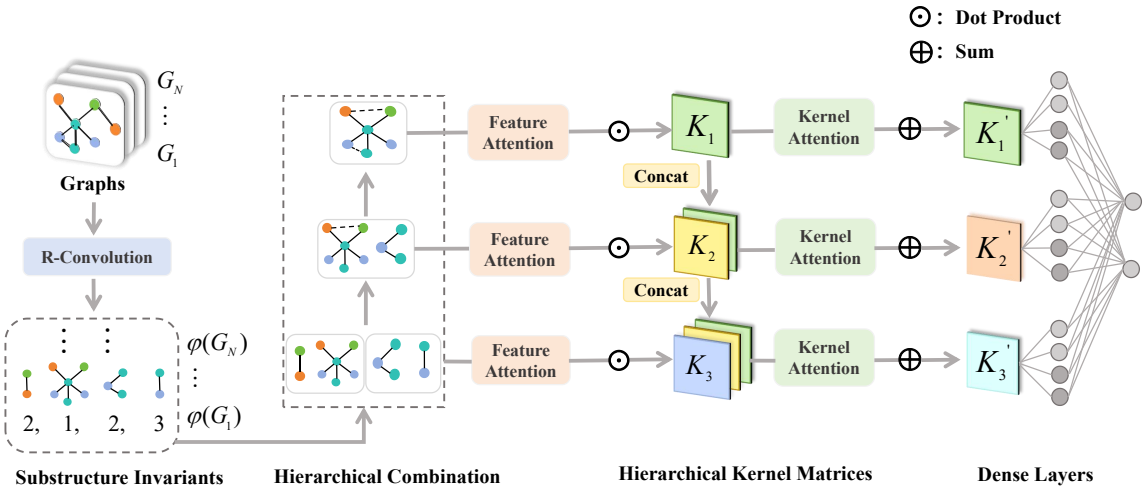


Figure 2: The framework of the proposed DHAKR model.

The Proposed DHAKR Model

The Overall Framework of the DHAKR Model

The overall framework is exhibited in Fig.2. Specifically, it contains five main computational procedures. **First**, we compute the feature vector $\phi(G_i)$ for each graph G_i based on a specific R-convolution kernel. Since the WLSK and SPGK have more effective performance for graph classification and their associated substructures have proven to be powerful structural characteristics of the original graph, in this work we extract the shortest paths and WL-based subtrees for $\phi(G_i)$. **Second**, we hierarchically combine original substructures into more meaningful structures, by adaptively computing the assignment matrix to hierarchically converge the homogeneous substructure invariants into composite substructure invariants. **Third**, during the above combination process, we employ the attention mechanism to discriminate the importance of different substructures for the substructure invariant converging. **Fourth**, we compute the hierarchical kernel matrices by concatenating the individual kernel matrices based on different hierarchical-level composite substructure invariants. Each individual kernel matrix is assigned attention-based distinct weights for the concatenation. **Finally**, we employ the above concatenated kernel matrices as the multi-scale kernel-based graph embedding representations for the classifier.

The Detailed Definitions of the DHAKR Model

The Construction of Substructure Invariants. We employ the WLSK and SPGK kernels to extract the WL subtrees and shortest paths as the initial substructure invariants. For the DHAKR(WL) model, the subtree-based feature vector $\phi_{\text{WL}}(G)$ of graph G is defined as

$$\phi_{\text{WL}}(G) = [c(G, l_1), c(G, l_2), ..., c(G, l_{|\mathcal{L}|})], \quad (5)$$

where l_i is the node label defined by Eq.(2), $|\mathcal{L}|$ refers to the number of all possible WL subtree invariants and each element $c(G, l_i)$ corresponds to the number of the subtree labeled by l_i in G. Similarly, for the DHAKR(SP) model, the shortest path-based feature vector $\phi_{\text{SP}}(G)$ is defined as

$$\phi_{\text{SP}}(G) = [c(G, p_1), c(G, p_2), ..., c(G, p_{|\mathcal{P}|})], \quad (6)$$

where $c(G, p_i)$ denotes the number of the shortest paths of length p_i in G, and $|\mathcal{P}|$ indicates the number of all possible shortest path. With $\phi_{\text{WL}}(G)$ or $\phi_{\text{SP}}(G)$ to hand, we derive the feature matrix $\mathbf{X}_{(\cdot)}$ for the entire dataset $\mathbf{G}$ as

$$\mathbf{X}_{(\cdot)} = \begin{pmatrix} \phi_{(\cdot)}(G_1) \\ ... \\ \phi_{(\cdot)}(G_i) \\ ... \\ \phi_{(\cdot)}(G_N) \end{pmatrix}, \quad (7)$$

where $G_i \in \mathbf{G}$, and $(\cdot)$ corresponds to either the symbol WL for WLSK or the symbol SP for SPGK.

The Hierarchical Combination of Substructures. We propose the H-hierarchical combination of substructure invariants to capture the dependence and homogeneity between the invariants. With the feature matrix $\mathbf{X}_{(\cdot)}^{(h)} \in \mathbb{R}^{N \times f_h}$ at layer h, we compute the soft cluster assignment matrix as

$$\mathbf{S}^{(h)} = softmax(\mathbf{X}^{(h)^T}\mathbf{W} + b) \in \mathbb{R}^{f_h \times f_{h+1}}, \quad (8)$$

where $\mathbf{W}$ and b are the weight and bias in the linear neural network, and f_h denotes the feature dimensions at layer h. $\mathbf{S}^{(h)}$ represents the probability of substructures from layer h being assigned to each cluster at layer $h+1$. This allows us to combine substructure invariants in a flexible manner, where substructures with high relevance are more likely to be grouped together. Thus, we can derive the feature matrix $\mathbf{X}^{(h+1)}$ at layer $h+1$ as

$$\mathbf{X}_{(\cdot)}^{(h+1)} = \mathbf{X}_{(\cdot)}^{(h)}\mathbf{S}^{(h)} \in \mathbb{R}^{N \times f_{h+1}}. \quad (9)$$

The Attention Mechanism for Feature Selection. During the hierarchical combination procedure, since the importance of each substructure is different, we introduce the

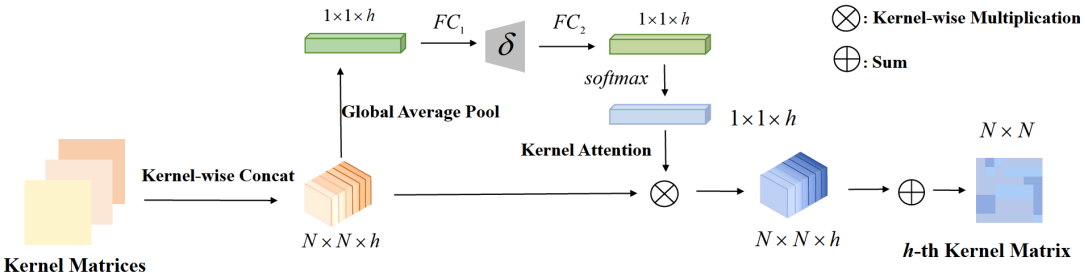


Figure 3: An instance of the construction of the h-th layer weighted kernel matrix.

attention mechanism to facilitate feature selection, aiming to eliminate the redundant substructure information unsuitable for graph classification. Inspired by the channel attention (Hu, Shen, and Sun 2018), we employ the *Average Pooling* (AP) operation to squeeze the representation into a feature channel. With the feature matrix $\mathbf{X}^{(h)}$ at layer h to hand, we derive the aggregation information $\mathbf{e}^h \in \mathbb{R}^{1\times f_h}$ as

$$e_k^h = F_{AP}(X_{(\cdot)}^{(h)}(:,k)) = \frac{1}{N}\sum_{i=1}^{N} x_{i,k}^h, \tag{10}$$

where e_k^h is the k-th element of $\mathbf{e}^h$. The resulting attention scores of different substructures are computed through two fully-connected layers with the activation function σ as

$$\boldsymbol{\alpha}^h = softmax(W_2\sigma(W_1\mathbf{e^h})), \tag{11}$$

where W_1 and W_2 are learnable weights. $\boldsymbol{\alpha}^h$ represents the attention score for each substructure invariant. With the attention scores, the feature matrix $\mathbf{X}^{(h)}$ can be updated as the attention-based weighted feature matrix $\mathbf{X}_{(\cdot)}^{(h)'}$, i.e.,

$$\mathbf{X}_{(\cdot)}^{(h)'} = \boldsymbol{\alpha}^h \otimes \mathbf{X}_{(\cdot)}^{(h)}, \tag{12}$$

where $\otimes$ indicates the feature-wise multiplication. By replacing $\mathbf{X}_{(\cdot)}^{(h)'}$ of Eq.(10) with that of Eq.(12), we can compute the weighted composite substructure invariants.

The Construction of Hierarchical Kernel Matrices. Based on the definition in (Haussler et al. 1999), the kernel matrix $\mathbf{K}_h$ can be computed directly by the dot product of pairwise feature vectors, i.e.,

$$\mathbf{K}_h = \mathbf{X}_{(\cdot)}^{(h)'} \cdot \mathbf{X}_{(\cdot)}^{(h)'^T}. \tag{13}$$

When we vary the parameter h from 1 to H, a family of H-hierarchical kernel matrices is formed as

$$\mathbb{K}_H = \{\mathbf{K}_1, ..., \mathbf{K}_h, ..., \mathbf{K}_H\}, \tag{14}$$

To capture complex dependencies among the kernel matrices and extract multi-scale kernel embeddings, we employ channel attention to assign the kernel matrices different weights. We treat the number of kernel matrices as the number of channels, i.e., concatenate h kernel matrices into a 3D matrix $\mathcal{K}_h \in \mathbb{R}^{N\times N\times h}$ with h channels. We first use the *Global Average Pooling* (GAP) to extract the aggregation information $\mathbf{e}^{h'} \in \mathbb{R}^{1\times h}$ in the kernel channel, i.e.,

$$e_j^{h'} = F_{GAP}(\mathcal{K}_h(:,:,j)) = \frac{1}{N}\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{N} K_h(i,k,j), \tag{15}$$

where $e_j^{h'}$ is the j-th element of $\mathbf{e}^{h'}$. Similar to the feature attention, we also use two fully-connected layers with activation function as shown in Eq.(11). Then, the kernel attention score $\boldsymbol{\alpha}^{h'}$ is derived. Different from the feature attention of Eq.(12), we use the kernel-wise multiplication and summation for the hierarchical kernel matrices from layer 0 to h, and update the h-th layer weighted kernel matrix as

$$\mathbf{K}'_h = \sum_{i=1}^{h} \boldsymbol{\alpha}^{i'} \otimes \mathbf{K}_i \tag{16}$$

An instance of the construction of the h-th layer weighted kernel matrix is shown in Fig.3. Hence, the family of H-hierarchical kernel matrices can be updated as

$$\mathbb{K}'_H = \{\mathbf{K}'_1, ..., \mathbf{K}'_h, ..., \mathbf{K}'_H\}. \tag{17}$$

The Kernel-based Graph Embedding Representations. We employ the kernel matrices as graph embedding representations. (Bunke and Riesen 2008) and (Bai, Hancock, and Han 2013) have demonstrated that a graph can be embedded into a feature vector by means of its (dis)similarities to prototype graphs. Inspired by this method, we employ all sample graphs as prototype graphs and embed each graph into vectors using the kernel-based graph similarities. The resulting kernelized graph embeddings at layer h is

$$\phi_{(\cdot)}^h(G) = [K'_{h(\cdot)}(G,G_1), ..., K'_{h(\cdot)}(G,G_i), ..., K'_{h(\cdot)}(G,G_N)], \tag{18}$$

where each element $K'_{h(\cdot)}(G,G_i)$ denotes the kernel value between G and G_i at layer h, and $(\cdot)$ represents either the WLSK or SPGK kernel.

Objective Functions. To introduce the supervision signals to guide the model, we feed the family of H-hierarchical kernel embeddings $\mathbb{K}'_H$ to dense layers for classification. Given N graphs and M labels, we can derive the label prediction $\hat{\mathbf{Y}} \in \mathbb{R}^{N\times M}$ in the following way:

$$\hat{\mathbf{Y}} = softmax(\sum_{h=1}^{H} \mathbf{W_h} \cdot \mathbf{K}'_h + \mathbf{b_h}), \tag{19}$$

Datasets	MUTAG	PTC(MR)	PROTEINS	IMDB-B	IMDB-M	Shock
Max # vertices	28	109	620	136	89	33
Mean # vertices	17.93	25.56	39.06	19.77	13	13.16
# graphs	188	344	1113	1000	1500	150
# classes	2	2	2	2	3	10
Description	Bio	Bio	Bio	SN	SN	CV

Table 1: Information of the graph datasets

where $\mathbf{W_h}$ and $\mathbf{b_h}$ are the weight and bias at the h-th layer. Then the cross-entropy loss for graph classification over all training graphs is represented as L_C:

$$L_C = -\sum_{i=1}^{N}\sum_{j=1}^{M} y_i log(\hat{y}_{i,j}), \tag{20}$$

where y_i denotes the real label of the i-th graph. Note that, each substructure should be assigned to a specific cluster, i.e., the cluster assignment probability for each substructure should be close to a one-hot vector. Thus, we add the regulation loss for soft cluster assignment matrix $\mathbf{S}$:

$$L_E = \frac{1}{H}\frac{1}{R}\sum_{h=1}^{H}\sum_{r=1}^{R} H_E(S_{r,\cdot}^{(h)}), \tag{21}$$

where $S_{r,\cdot}^{(h)}$ denotes the r-th row of $\mathbf{S}$ at the h-th layer. H_E indicates the entropy function. In summary, we have the following overall objective function:

$$loss = L_C + \gamma L_E, \tag{22}$$

where γ is the coefficient of cluster assignment regulation.

Computational Complexity Analysis

With $|V|$ and $|E|$ as the average number of nodes and edges, N is the number of graphs, T_{max} is the number of iterations in WLSK, d is the largest feature dimension, and H is the number of hierarchical combined layers. First, the time complexities of computing explicit feature vectors based on SPGK and WLSK are $O(N|V|^3 + Nd^2)$ and $O(T_{max}|E|)$ respectively. Besides, the time complexities of feature attention and kernel attention are $O(Nd + d^2)$ and $O(N^2H + H^2)$. The time complexities of computing assignment loss and classifier are both $O(N^2d)$. Since $H << N$ and $H << d$, the total computational complexities of DHAKR(SP) and DHAKR(WL) are $O(N|V|^3 + Nd^2 + N^2d)$ and $O(T_{max}|E| + d^2 + N^2d)$.

Discussions of the Proposed DHAKR Model

Unlike some existing state-of-the-art methods, the proposed DHAKR has some theoretical advantages. **First**, the proposed DHAKR can either utilize the independent substructures to capture the local information or employ the composite substructures to extract the global structure similarity. **Second**, we integrate the feature attention mechanism to assign larger weights to more significant substructures. Additionally, the end-to-end framework allows us to adaptively identify more important structural features. **Third**, the proposed DHAKR model can identify the common structural patterns by computing the kernel-based similarity over all sample graphs.

Experiments

We evaluate the classification performance of the DHAKR model on standard graph datasets(Siddiqi et al. 1999; Morris et al. 2020) extracted from bioinformatics (Bio), social networks (SN), and computer vision (CV). The statistical information of the datasets is shown in Table 1.

Comparisons with Graph Kernels

Experimental Settings. We compare the proposed DHAKR with six existing graph kernels, including 1) GCGK (Shervashidze et al. 2009) 2) RWGK (Gärtner, Flach, and Wrobel 2003), 3) SPGK (Borgwardt and Kriegel 2005), 4) WLSK (Shervashidze et al. 2011), (5) CORE SP (Nikolentzos et al. 2018a), and (6) CORE WL (Nikolentzos et al. 2018b). To make a fair comparison, we perform a 10-fold cross-validation and repeat the experiments 10 times. Table 2 reports the average accuracies and standard deviations. Since some datasets are not evaluated in the original papers, we do not report the results. We use DHAKR(WL) and DHAKR(SP) to denote the DHAKR variants based on WLSK and SPSK. For our proposed methods, the assignment ratio is 0.5. The detailed descriptions of baselines and the implementation details are provided in the Arxiv version.

Experimental Results and Analysis. As we can observe from Table 2, our proposed DHAKR model achieves highly competitive performance. Compared to the state-of-the-art graph kernels, DHKAR has the highest accuracies except for PROTEINS. Note that, DHAKR(SP) still performs better than the original SPGK on PROTEINS. These observations demonstrate the theoretical advantages of the proposed DHAKR, i.e., adaptively identifying the importance of different substructures and hierarchically integrating the relationships between substructure invariants. The experimental results also indicate that the end-to-end kernel learning framework is more beneficial for classification.

Comparisons with Graph Deep Learning

Experimental Settings. We compare the proposed DHAKR with eight alternative graph deep learning methods, including 1) DGCNN (Zhang et al. 2018), 2) DCNN (Atwood and Towsley 2016), 3) PATCHY-SAN (Niepert, Ahmed, and Kutzkov 2016), 4) DGK (Yanardag and Vishwanathan 2015), 5) q-RWNN (Nikolentzos and Vazirgiannis 2020) associated with three different random walk length q ($q = 1, 2, 3$) 6) KerGNN (Feng et al. 2022) 7) CapsGNN (Xinyi and Chen 2019), and 8) GIN (Xu et al. 2019).

Datasets	MUTAG	PTC(MR)	PROTEINS	IMDB-B	IMDB-M	Shock	A.R.
GCGK3	82.04±0.39	55.41±0.59	71.67±0.55	–	–	26.93±0.63	7.6
RWGK	80.77±0.72	55.91±0.37	74.20±0.40	67.94±0.77	46.72±0.30	2.31±1.13	6.8
SPGK	83.38±0.81	55.52±0.46	75.10±0.50	71.26±1.04	51.33±0.57	37.88±0.93	5.2
WLSK	82.88±0.57	58.26±0.47	73.52±0.43	71.88±0.77	49.50±0.49	36.40±1.00	5.5
CORE SP	88.29 ±1.55	59.06 ±0.93	–	72.62 ±0.59	49.43±0.42	–	6.6
CORE WL	87.47±1.08	59.43±1.20	–	74.02±0.42	51.35±0.48	–	4.8
DHAKR(SP)	**88.44± 0.64**	**67.33±1.52**	76.09±0.97	**76.11±0.61**	**53.21±0.54**	**55.41±1.67**	**1.5**
DHAKR(WL)	**89.87±1.03**	**68.76±0.96**	**77.47±0.53**	**75.21±0.54**	**52.07±0.39**	**48.75±1.26**	**1.7**

Table 2: Classification accuracy (in % ± standard error) comparisons with graph kernels. A.R. is the Average Rank.

Datasets	MUTAG	PTC(MR)	PROTEINS	IMDB-B	IMDB-M	A.R.
DGCNN	85.83±1.66	58.57±1.69	75.54±0.94	70.03±0.86	47.83±0.85	7.0
DCNN	66.98	56.60	61.29±1.60	49.06±1.37	46.72±0.30	10.8
PATCHY-SAN	88.95±4.37	62.29 ± 5.68	75.00±2.51	71.00±2.29	45.23±2.84	5.8
DGK	82.66±1.45	60.08 ± 2.55	71.68±0.50	66.96±0.56	44.55±0.52	9.4
1-RWNN	89.2±4.3	57.36±1.66	70.8±4.8	70.8±4.8	47.8±3.8	7.4
2-RWNN	88.1±4.8	57.61±1.43	74.7±3.3	70.6±4.4	48.8±2.9	6.8
3-RWNN	88.6±4.1	58.40±1.86	74.1±2.8	70.7±3.9	47.8±3.5	7
KerGNN	–	–	76.5±3.9	74.4±4.3	51.6±3.1	8.2
CapsGNN	86.67±6.88	–	76.28±3.63	73.10±4.83	50.27±2.65	5.6
GIN	84.7±6.7	64.29±1.26	74.3±3.3	71.23±3.9	48.53±3.3	6.0
DHAKR(SP)	88.44± 0.64	**67.33±1.52**	76.09±0.97	**76.11±0.61**	**53.21±0.54**	**2.4**
DHAKR(WL)	**89.87±1.03**	**68.76±0.96**	**77.47±0.53**	**75.21±0.54**	**52.07±0.39**	**1.4**

Table 3: Classification accuracy (in % ± standard error) comparisons with deep learning methods. AR is the Average Rank.

Experimental Results and Analysis. Table 3 indicates that the proposed DHAKR outperforms graph deep learning methods on all datasets. DHAKR(SP) achieves the best classification performance on PTC(MR), IMDB-BINARY and IMDB-MULTI datasets. DHAKR(WL) also has the highest accuracy on MUTAG and PROTEINS datasets. The experimental results meet our expectations since the pooling operation discards the graph information for existing GNNs. Besides, these results also demonstrate the effectiveness of the kernel-based framework, i.e., considering the common patterns shared between all graphs instead of stacking more message-passing layers.

The Further Analysis for DHAKR

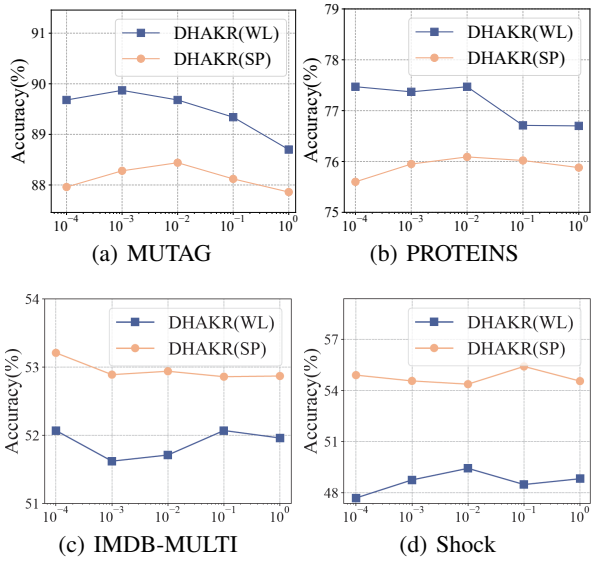
The Ablation Study. In this section, we compare DHAKR with its variants on three datasets to validate the effectiveness of the feature attention and kernel attention components. From the results shown in Table 4, we can draw the following conclusions:(1) Whether based on SPGK or WLSK, the accuracies of DHAKR are better than models without kernel attention, verifying the usefulness of kernel attention. (2) When the model removes the feature attention component, the classification accuracy drops sharply, demonstrating the effectiveness of adaptively identifying the importance of substructure invariants. The additional results are shown in the Arxiv version.

Hyperparameter Analysis. We further investigate the sensitivity of γ in Eq.(22). We vary the values of γ from 0.0001 to 1.0 and test the graph classification performance on four datasets as shown in Fig.4. With the increase of the regularization coefficient γ, the classification accuracies rise first and drop slowly on MUTAG and PROTEINS datasets, whereas it initially drops and then gradually improves on the IMDB-MULTI and Shock datasets. It indicates that different values of γ may result in varying classification performances. Fig.4 also shows that the regularization coefficient is more sensitive in the DHAKR(WL) model, i.e., it has a higher impact on graph classification accuracy compared to the DHAKR(SP) model. We also report the results of hyperparameter analysis on PTC(MR) and IMDB-BINARY datasets in the Arxiv version.

Datasets	MUTAG	PROTEINS	IMDB-M
DHAKR(WL)	**89.87±1.03**	**77.47±0.53**	**52.07±0.39**
DHAKR(WL) w/o KAtt	89.62±0.79	77.05±0.33	51.87±0.60
DHAKR(WL) w/o FAtt	86.81±1.29	72.74±0.96	49.39±0.79
DHAKR(SP)	**88.44±0.64**	**76.09±0.97**	**53.21±0.54**
DHAKR(SP) w/o KAtt	88.33±0.59	75.79±1.22	52.50±0.43
DHAKR(SP) w/o FAtt	80.05±2.40	74.55±0.51	44.36±0.51

Table 4: The ablation study of DHAKR. w/o KAtt means variants without kernel attentions and w/o FAtt denotes variants without feature attentions.

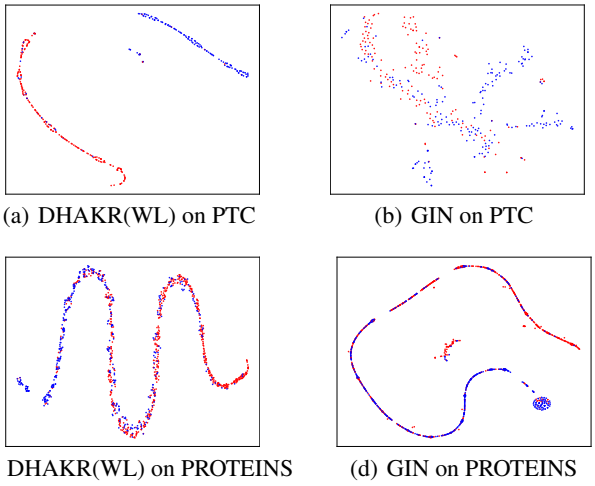
Visualization of Graph Representations. To intuitively demonstrate the effectiveness of our proposed method in learning graph representation, we conduct the visualization using t-SNE (Van der Maaten and Hinton 2008) and compare the representations with the GIN backbone on the PTC(MR) and PROTEINS datasets as Fig.5. For the DHAKR(WL) model, we extract the graph embeddings before dense layers for classifying. For the GIN model, we extract the representations by the readout function. The results



(a) MUTAG (b) PROTEINS

(c) IMDB-MULTI (d) Shock

Figure 4: The hyper-parameter sensitivity study.



(a) DHAKR(WL) on PTC (b) GIN on PTC

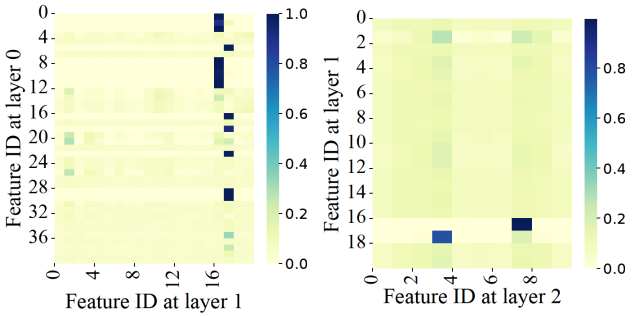
(c) DHAKR(WL) on PROTEINS (d) GIN on PROTEINS

Figure 5: The t-SNE visualization of graph representations.

in Fig.5 show that DHAKR provides more distinct boundaries between graphs of different classes while effectively clustering graphs of the same class together.

Visualization of Hierarchical Combination. We further visualize the assignment score heatmaps for the MUTAG dataset based on the DHAKR(WL) model. Since the DHAKR model has a three-layer hierarchical structure with two assignment matrices, we provide corresponding visualizations of the matrix for each layer. The results in Fig.6 demonstrate that DHAKR can adaptively integrate homogeneous substructure invariants to generate new features.

To present the process of hierarchical feature assignment more intuitively, we select 24 substructures from the MUTAG dataset and assign them to the cluster with the highest probability after 500 training epochs. The visualization of



(a) Hierarchical Layer 1 (b) Hierarchical Layer 2

Figure 6: The assignment score visualization on MUTAG.

hierarchical composite features in Fig.7 indicates that homogeneous substructures tend to cluster together. Therefore, the combined hierarchical framework enables DHAKR to capture global topological information at higher layers and local substructures at lower layers.

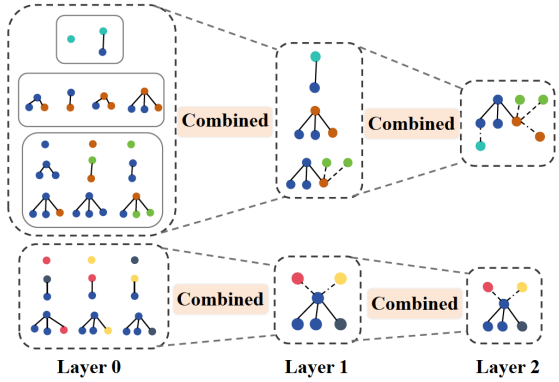


Figure 7: The hierarchical composite feature visualization.

Conclusion

In this paper, we have proposed a novel DHAKR model for graph classification. This model cannot only hierarchically extract composite substructure invariants, but also adaptively identify the importance of different substructures. Moreover, the proposed model can provide an end-to-end learning architecture that captures the common structural patterns over all sample graphs. Experiments demonstrate the superior classification performance.

Our future work is to consider the hybrid informative substructures, including random walks, cycles, etc. Moreover, the structurally aligned (sub)structures, as successfully explored in our previous works (Bai et al. 2024; Cui et al. 2024; Bai et al. 2015), will also be further utilized for proposing novel aligned kernelized methods. Finally, we will also utilize the subtrees extracted from the Perron-Frobenius operator of hypergraphs (Bai, Ren, and Hancock 2014; Bai, Escolano, and Hancock 2016), resulting in novel hypergraph-based kernalized methods.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants T2122020, 61602535, 61976235. Ming Li acknowledged the supports from the Pioneer and Leading Goose R&D Program of Zhejiang (No. 2024C03262), the National Natural Science Foundation of China (No. U21A20473, No. 62172370) and the Jinhua Science and Technology Plan (No. 2023-3-003a). Hangyuan Du acknowledged the support from the Humanity and Social Science Foundation of Ministry of Education (24YJAZH022). This work is also supported in part by the Program for Innovation Research in the Central University of Finance and Economics, and the Emerging Interdisciplinary Project of CUFE.

References

An, J.; Qu, C.; Zhou, Z.; Cao, F.; Xu, Y.; Qi, Y.; and Shen, F. 2024. Hybrid Directional Graph Neural Network for Molecules. In *Proceedings of ICLR*. OpenReview.net.

Atwood, J.; and Towsley, D. 2016. Diffusion-convolutional Neural Networks. *Proceedings of NeurIPS*, 29.

Bai, L.; Cui, L.; Wang, Y.; Li, M.; Li, J.; Yu, P. S.; and Hancock, E. R. 2024. HAQJSK: Hierarchical-Aligned Quantum Jensen-Shannon Kernels for Graph Classification. *IEEE Trans. Knowl. Data Eng.*, 36(11): 6370–6384.

Bai, L.; Escolano, F.; and Hancock, E. R. 2016. Depth-based Hypergraph Complexity Traces from Directed Line Graphs. *Pattern Recognit.*, 54: 229–240.

Bai, L.; Hancock, E. R.; and Han, L. 2013. A Graph Embedding Method using the Jensen-Shannon Divergence. In *Proceedings of CAIP*, 102–109.

Bai, L.; Ren, P.; and Hancock, E. R. 2014. A Hypergraph Kernel from Isomorphism Tests. In *Proceedings of ICPR*, 3880–3885. IEEE Computer Society.

Bai, L.; Zhang, Z.; Wang, C.; Bai, X.; and Hancock, E. R. 2015. A Graph Kernel Based on the Jensen-Shannon Representation Alignment. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of IJCAI*, 3322–3328.

Borgwardt, K. M.; and Kriegel, H.-P. 2005. Shortest-path Kernels on Graphs. In *Proceedings of ICDM*, 8–pp. IEEE.

Bunke, H.; and Riesen, K. 2008. Graph Classification Based on Dissimilarity Space Embedding. In *Proceedings of SPR and SSPR*, volume 5342 of *Lecture Notes in Computer Science*, 996–1007.

Cui, L.; Bai, L.; Bai, X.; Wang, Y.; and Hancock, E. R. 2024. Learning Aligned Vertex Convolutional Networks for Graph Classification. *IEEE Trans. Neural Networks Learn. Syst.*, 35(4): 4423–4437.

Feng, A.; You, C.; Wang, S.; and Tassiulas, L. 2022. Kergnns: Interpretable Graph Neural Networks with Graph Kernels. In *Proceedings of AAAI*, 6614–6622.

Gärtner, T.; Flach, P.; and Wrobel, S. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Proceedings of COLT*, 129–143. Springer.

Haussler, D.; et al. 1999. Convolution Kernels on Discrete Structures. Technical report, Citeseer.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation Networks. In *Proceedings of CVPR*, 7132–7141.

Kelvinius, F. E.; Georgiev, D.; Toshev, A. P.; and Gasteiger, J. 2023. Accelerating Molecular Graph Neural Networks via Knowledge Distillation. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Proceedings of NeurIPS*.

Kosmala, A.; Gasteiger, J.; Gao, N.; and Günnemann, S. 2023. Ewald-based Long-Range Message Passing for Molecular Graphs. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of ICML*, volume 202, 17544–17563. PMLR.

Kriege, N. M.; Giscard, P.-L.; and Wilson, R. 2016. On valid Optimal Assignment Kernels and Applications to Graph Classification. *Proceedings of NeurIPS*, 29.

Kriege, N. M.; and Mutzel, P. 2012. Subgraph Matching Kernels for Attributed Graphs. In *Proceedings of ICML*. icml.cc / Omnipress.

Maleki, N.; Padmanabhan, B.; and Dutta, K. 2022. Representing Social Networks as Dynamic Heterogeneous Graphs. In *Proceedings of ICDM*, 1–10. IEEE.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. Tudataset: A Collection of Benchmark Datasets for Learning with Graphs. *arXiv preprint arXiv:2007.08663*.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning Convolutional Neural Networks for Graphs. In *Proceedings of ICML*, 2014–2023.

Nikolentzos, G.; Meladianos, P.; Limnios, S.; and Vazirgiannis, M. 2018a. A Degeneracy Framework for Graph Similarity. In *Proceedings of IJCAI*, 2595–2601.

Nikolentzos, G.; Meladianos, P.; Limnios, S.; and Vazirgiannis, M. 2018b. A Degeneracy Framework for Graph Similarity. In *Proceedings of IJCAI*, 2595–2601.

Nikolentzos, G.; and Vazirgiannis, M. 2020. Random Walk Graph Neural Networks. In *Proceedings of NeurIPS*.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman Graph Kernels. *Journal of Machine Learning Research*, 12(9).

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *Artificial intelligence and statistics*, 488–495. PMLR.

Siddiqi, K.; Shokoufandeh, A.; Dickinson, S. J.; and Zucker, S. W. 1999. Shock Graphs and Shape Matching. *International Journal of Computer Vision*, 35: 13–32.

Togninalli, M.; Ghisu, E.; Llinares-López, F.; Rieck, B.; and Borgwardt, K. 2019. Wasserstein weisfeiler-lehman graph kernels. *Proceedings of NeurIPS*, 32.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of machine learning research*, 9(11).

Wei, C.; Liang, J.; Liu, D.; and Wang, F. 2022. Contrastive Graph Structure Learning via Information Bottleneck for Recommendation. In Koyejo, S.; Mohamed, S.; Agarwal,

A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Proceedings of NeurIPS*.

Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; and Tan, T. 2019. Session-based Recommendation with Graph Neural Networks. In *Proceedings of AAAI*, 346–353.

Xinyi, Z.; and Chen, L. 2019. Capsule Graph Neural Network. In *Proceedings of ICLR*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *Proceedings of ICLR*. OpenReview.net.

Yanardag, P.; and Vishwanathan, S. 2015. Deep Graph Kernels. In *Proceedings of ACM SIGKDD*, 1365–1374.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-end Deep Learning Architecture for Graph Classification. In *Proceedings of AAAI*.