DHTAGK: Deep Hierarchical Transitive-Aligned Graph Kernels for Graph Classification

Xinya Qin[1*], **Lu Bai**[1*], **Lixin Cui**[2], **Ming Li**[3†], **Ziyu Lyu**[4], **Hangyuan Du**[5], **Edwin Hancock**[6]

[1]School of Artificial Intelligence, 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]School of Cyber Science and Technology, Sun Yat-Sen University, Shenzhen, China
[5]School of Computer and Information Technology, Shanxi University, Taiyuan, China
[6]Department of Computer Science, University of York, York, United Kingdom.

Abstract

In this paper, we propose a family of novel Deep Hierarchical Transitive-Aligned Graph Kernels (DHTAGK) for graph classification. To this end, we commence by developing a new Hierarchical Aligned Graph Auto-Encoder (HA-GAE) to construct transitive-aligned embedding graphs that encapsulate the structural correspondence information between graphs. The DHTAGK kernels then measure either the Jensen-Shannon Divergence between the adjacency matrices or the Gaussian kernel between the node feature matrices of the embedding graphs. Unlike the classical R-convolution kernels and node-based alignment kernels, the DHTAGK kernels can capture the transitive structural correspondence information and thus ensure the positive definiteness. Furthermore, the HA-GAE enables the DHTAGK kernels to simultaneously reflect both local and global graph structures and identify common structural patterns. Experimental results show that the DHTAGK kernels outperform state-of-the-art graph kernels and deep learning methods on benchmark datasets.

1 Introduction

Graph-based machine learning is a crucial research area in artificial intelligence, with wide applications in fields such as social network analysis, bioinformatics, and computer vision. However, the complexity and irregularity of graph data present significant challenges for traditional machine learning algorithms, which are usually designed for vectorial data rather than graph structures. Thus, there is a need for methods that can effectively learn numerical features from graphs.

One effective approach to learning numerical features from graphs is to employ graph kernels, that define a similarity measure between graphs. Graph kernels can map graphs into a high-dimensional Hilbert space, capturing their structural information more effectively. Currently, most existing graph kernels are developed based on the R-convolution framework proposed by [Haussler, 1999]. This framework defines graph kernels by decomposing graphs into substructures and counting the number of isomorphic substructures shared between pairwise graphs. Under this foundation, numerous graph kernels have been developed (see more details in Section 2.1). Despite their effectiveness, the R-convolution graph kernels still suffer from several challenges.

First, the R-convolution graph kernels ignore the positional correspondence information between isomorphic substructures, which leads to unreliable kernel measures between graphs. For instance, the Random Walk Graph Kernel (RWGK) [Gartner *et al.*, 2003], which is an early typical work, only focuses on counting the number of random walks with same lengths, even many walks are not structurally aligned to each other in terms of the global graph structures. This drawback may influence the effectiveness of some applications, including graph-based bioinformatics, where isomorphic molecular motifs located in different regions may exhibit different trait expressions. In recent years, many studies have explored correspondence information in graphs, such as [Bai *et al.*, 2015a; Bai *et al.*, 2019; Bai *et al.*, 2022; Cui *et al.*, 2024]. Among them, [Bai *et al.*, 2015a] have proposed an Aligned Subtree Kernel (ASK), that can effectively explore the node-based matching information between the rooted nodes of subtrees. Unfortunately, the node correspondences identified by this kernel are not transitive (i.e., for three nodes a, b and c, if a matches with b, and b matches with c, it does not guarantee that a matches with c). Thus, this kernel is not positive definite (pd).

Moreover, the R-convolution graph kernels mainly capture the local information from the substructures of small sizes, thereby neglecting the global topological information. For instance, the Graphlet Kernel (GK) [Shervashidze *et al.*, 2009] only considers graphlets with 3-5 nodes, which may overlook important global topological features of graphs. To overcome

*XinyaQin@mail.bnu.edu.cn, bailu@bnu.edu.cn
†Corresponding Author: mingli@zjnu.edu.cn

this limitation, a family of global graph kernels have been developed to focus more on the global structural graph characteristics through the adjacency matrices. For instance, [Johansson *et al.*, 2014] have developed a Global Geometric Embedding Kernel based on the Lovász numbers as well as their orthonormal representations, that are computed through the adjacency matrix. [Xu *et al.*, 2018] have proposed a Global Reproducing Graph Kernel by measuring the similarity between the approximated von Neumann entropies based on the adjacency matrix. However, these kernels can only capture global similarities, lacking detailed structural information residing in the internal topological structure of graphs.

Finally, the R-convolution graph kernels only focus on the similarity measure between each individual pair of graphs, and thus cannot capture the common structural pattern information shared over all sample graphs. Overall, the above analysis indicates that developing effective graph kernels is still a challenging problem.

This work aims to overcome the above theoretical limitations of the existing R-convolution kernels by developing a family of novel Deep Hierarchical Transitive-Aligned Graph Kernels (DHTAGK). One key innovation of the new kernels is the construction of the Hierarchical Aligned Graph Auto-Encoder (HA-GAE), which not only hierarchically extracts a series of transitive-aligned embedding graphs to capture both the global and local structure information, but also captures the common hierarchical structural pattern information over all graphs. Overall, our contributions are threefold.

First, we develop a new HA-GAE to hierarchically capture the transitive node-level matching information between graphs. Specifically, for each encoding layer, we use the node assignment to group the input nodes into clusters and then compress the nodes within a cluster into a coarsened node, hierarchically transforming the original input graph into an embedding graph. For each decoding layer, we apply the node assignment to reconstruct the original graph structure by expanding each coarsened node into all retrieved nodes probabilistically. Since the HA-GAE is defined by employing the same node assignment function for all original graphs, the adjacency matrices and the node feature matrices of the embedding graphs extracted through the encoding layers are naturally transitive aligned. Furthermore, the HA-GAE can effectively capture the common structural patterns shared over all original graphs through the deep learning architecture.

Second, for pairwise graphs, we develop a new family of DHTAGK kernels based on the transitive-aligned adjacency matrices and node feature matrices of their embedding graphs constructed through the HA-GAE. Due to the theoretical properties of the HA-GAE, the proposed DHTAGK kernels are not only transitive aligned but also capture rich deep hierarchical structural characteristics of graphs, guaranteeing the positive definiteness and explaining the effectiveness for the proposed kernels (see details in Section 3.5).

Third, we evaluate the classification performance of the proposed kernels on standard graph datasets using the C-Support Vector Machine(C-SVM). Experimental results show that the proposed kernels outperform state-of-the-art graph kernels and graph deep learning methods.

This paper is organized as follows. Section 2 reviews some related works. Section 3 provides the definition of the proposed kernels. Section 4 presents experiments. Finally, Section 5 concludes this work.

2 Related Works

2.1 The Graph Kernels

The Graph kernels can provide an elegant way to map graphs into a high-dimensional Hilbert space, enabling linear algorithms to solve nonlinear problems in the original low-dimensional space. Most existing graph kernels are developed based on the R-convolution framework [Haussler, 1999], that decomposes graphs into substructures and counts the number of isomorphic substructures shared between pairwise graphs. This indicates that any graph decomposition method and substructure type can define a new graph kernel function. Under this principle scenario, the R-convolution graph kernels can be classified into the following three categories. **a) Random walk-based:** e.g., the RWGK [Gartner *et al.*, 2003], the marginalized kernel [Kashima *et al.*, 2003]; **b) Path-based:** e.g., the Shortest Path Graph Kernel (SPGK) [Borgwardt and Kriegel, 2005], the backtrackless kernel [Aziz *et al.*, 2013]; **c) Subtree/subgraph-based:** e.g., the Weisfeiler-Lehman Subtree Graph Kernel (WLSK) [Shervashidze *et al.*, 2011], the Graphlet Kernel (GK) [Shervashidze *et al.*, 2009]. Other R-convolution kernels also include the Segmentation Graph Kernel [Harchaoui and Bach, 2007], the Pyramid Quantized Weisfeiler-Lehman Kernel [Gkirtzou and Blaschko, 2016], the Quantum-inspired Jensen-Shannon Kernel [Bai *et al.*, 2020], etc.

Remark 1: Despite the performance of the R-convolution kernels for graph classification, they still suffer form several limitations. First, these kernels do not take the positional correspondence information between substructures into account. Second, these kernels tend to capture local information from small substructures, thereby neglecting the global topological information and thus overlooking important global topological features of graphs. Third, these kernels obtain kernel values through pairwise comparisons of graphs, lacking the ability to capture the common pattern information over all graph samples. In summary, these drawbacks limit the effectiveness of the existing R-convolution kernels.

2.2 The Graph Auto-encoders

Generally, the Graph Auto-Encoder (GAE) is developed by generalizing the classical Auto-Encoder into the graph domain [Kipf and Welling, 2016], and aims to learn a mapping function to convert node features into the deep latent space. Specifically, the encoder converts graphs into low-dimensional representations, while the decoder attempts to reconstruct the original graphs from the latent representations. Representative methods include a) the Adversarially Regularized Variational Graph Autoencoder (ARVGA) [Pan *et al.*, 2020] that employs a trainable scheme based on the Generative Adversarial Networks (GANs) [Goodfellow *et al.*, 2014], b) the Network Representations with Adversarially Regularized Autoencoders (NetRA) [Yu *et al.*, 2018] where the encoder and decoder are the LSTM networks [Hochreiter and Schmidhuber, 1997] associated with random walks

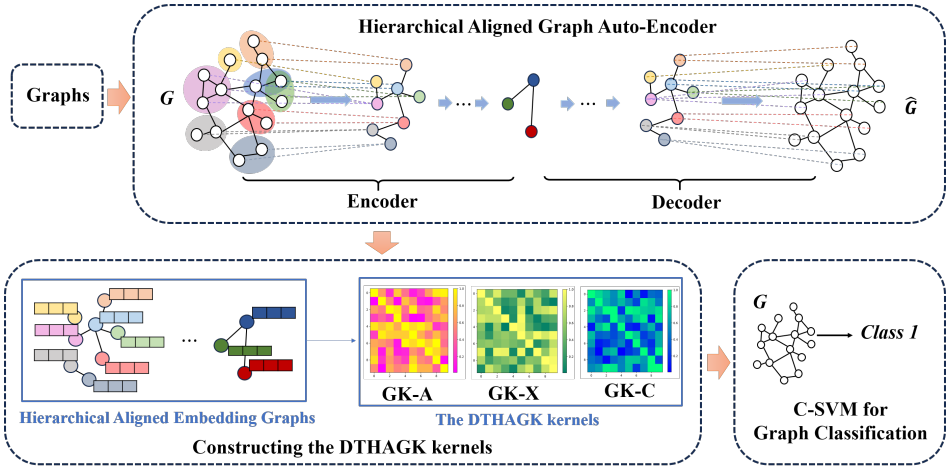

Figure 1: Our computational framework of the DHTAGK kernels.

rooted at each node as inputs, etc. Although the existing GAEs have shown their superior performance in link prediction, most of them tend to only reconstruct the structure information (i.e., the adjacency matrix) of graphs and ignore the restoration of the node feature matrix. Thus, these GAEs usually have lower performance for graph classification.

3 The Proposed DHTAGK Kernels

In this section, we provide the detailed definition of the proposed DHTAGK kernel associated with the newly developed HA-GAE model. Moreover, we highlight the theoretical advantages of the proposed kernels.

3.1 Preliminaries

We define a graph as $\mathrm{G}(A, X)$, where $A \in \{0,1\}^{n\times n}$ is the adjacency matrix, and $X \in \mathbf{R}^{n\times f}$ is the node feature matrix. In this work, node features are represented either by the one-hot encoding of node labels for attributed graphs or by node degrees for unattributed graphs.

To effectively process the graph data, Graph Neural Networks (GNNs) have been widely adopted. GNNs are deep learning models that generate node embeddings by integrating both node features and graph structure. These models typically aggregate the feature information from neighboring nodes to learn meaningful node representations. In this work, we employ the Graph Convolutional Network (GCN) layer [Kipf and Welling, 2017], which is defined as

$$\mathrm{GCN}(A, X) = \sigma\left(W\hat{A}X\right), \tag{1}$$

where $\hat{A} = I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, $\hat{A}$ is the adjacency matrix with self-loops, W is the trainable weight matrix, and σ is a non-linear activation function.

3.2 Frameworks of the DHTAGK Kernels

This subsection introduces the computational framework of the proposed DHTAGK, shown in Figure 1, consisting of three main computational steps. **First**, we propose a new HA-GAE model (see Section 3.3 for details), which is trained on all sample graphs and achieves the transitive-aligned node-level alignment. During the encoding, we hierarchically extract embedding graphs. **Second**, for pairwise graphs, we extract the transitive-aligned adjacency and node feature matrices of their embedding graphs through the HA-GAE. The DHTAGK kernels (kernel based on the adjacency Matrices (GK-A), kernel based on the node features (GK-X), and the composite kernel (GK-C)) are defined by measuring the kernel-based similarity between the embedding graphs (see Section 3.4 for details). **Third**, the kernel matrix is input into the C-SVM for graph classification.

3.3 The HA-GAE Model

The proposed HA-GAE model consists of an encoder and a decoder, which hierarchically reduces and restores the graph size. This process enables the capture of both local and global information across subgraphs of different sizes, addressing the limitations of the existing R-convolution graph kernels. Moreover, we train the HA-GAE model on a specific dataset, ensuring that all graphs in the dataset share the same trained node assignment function.

For an original sample graph, the encoder maps the node feature matrix and adjacency matrix to latent representations. Each encoder layer consists of two steps: *Assignment-E* and *Compression*. The *Assignment-E* step clusters nodes with similar latent semantics, while the *Compression* step combines nodes within the same cluster into a single coarsened node. The decoder mirrors the encoder's structure but the focus is on recovering the original node feature and adjacency matrices. Each decoder layer also involves two steps: *Assignment-D* and *Reconstruction*. This self-supervised learning approach maps the node feature and adjacency matrices into an embedding space, where effective structure representations are learned by minimizing reconstruction errors. Below, we provide a detailed definition of the encoder and the decoder for the HA-GAE model.

The Encoder of the HA-GAE

The architecture of the encoder is shown in Figure 2. The encoder maps input graphs into an embedding space, hier-

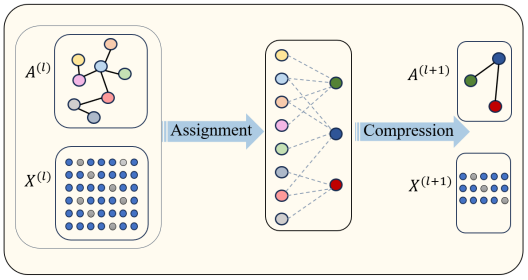

Figure 2: The specific framework of the l-th layer of the encoder.

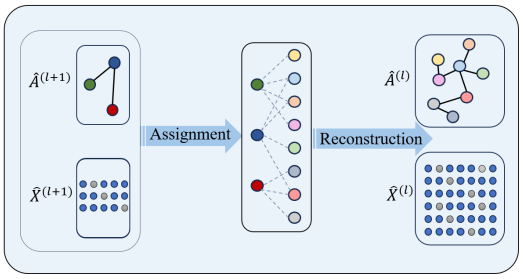

Figure 3: The specific framework of the l-th layer of the decoder.

archically generating a series of transitive-aligned embedding graphs. It consists of K layers. The original sample graph $G(A, X)$ serves as the input at the 0-th layer, represented as $G^{(0)}(A^{(0)}, X^{(0)})$. At the l-th layer ($l \in \{0, \ldots, K-1\}$), the encoder transforms the embedding graph $G^{(l)}(A^{(l)}, X^{(l)})$ into the $(l+1)$-th layer embedding graph $G^{(l+1)}(A^{(l+1)}, X^{(l+1)})$, where $A^{(l)} \in \mathbf{R}^{n_l \times n_l}$, $X^{(l)} \in \mathbf{R}^{n_l \times d_l}$, $A^{(l+1)} \in \mathbf{R}^{n_{l+1} \times n_{l+1}}$, $X^{(l+1)} \in \mathbf{R}^{n_{l+1} \times d_{l+1}}$. Here, $n_{(\cdot)}$ represents the number of nodes, and $d_{(\cdot)}$ represents the dimension of node features at each layer. Each encoder layer involves two operations: *Assignment-E* and *Compression*.

Assignment-E: In this step, nodes are assigned to different clusters. We compute the node assignment matrix $S^{(l)} \in \mathbf{R}^{n_l \times n_{l+1}}$ using a GNN model GNN_{assign} with input $(A^{(l)}, X^{(l)})$ followed by a Softmax function. The element $S^{(l)}_{i,j}$ represents the probability that node i at layer l is assigned to cluster j at layer $l+1$. The computation is defined as

$$S^{(l)} = softmax\left(GNN_{assign}\left(A^{(l)}, X^{(l)}\right)\right). \quad (2)$$

Compression: In this step, nodes belonging to the same cluster are compressed into a new coarsened node. First, we compute the node embedding matrix $Z^{(l)} \in \mathbf{R}^{n_l \times d_{l+1}}$ by a GNN layer GNN_{embed}, defined as

$$Z^{(l)} = GNN_{embed}\left(A^{(l)}, X^{(l)}\right). \quad (3)$$

Next, using the assignment matrix $S^{(l)}$, we compress the nodes by performing similar operations to the DiffPool [Ying *et al.*, 2018], i.e.,

$$A^{(l+1)} = S^{(l)\top} A^{(l)} S^{(l)}, \text{ and } X^{(l+1)} = S^{(l)\top} Z^{(l)}. \quad (4)$$

After K layers, the embedding graph $G^{(K)}(A^{(K)}, X^{(K)})$ from the final layer is passed to the decoder. By gradually reducing the number of clusters at each layer, the encoder helps abstract and aggregate node information into fewer nodes, thereby achieving a hierarchical transition from local to global information.

The Decoder of the HA-GAE

The architecture of the decoder is shown in Figure 3. The decoder reconstructs the original graph structure by probabilistically expanding each coarsened node into its corresponding nodes. The decoder also consists of K layers. At the l-th layer, it reconstructs the l-th embedding graph $\hat{G}^{(l)}(\hat{A}^{(l)}, \hat{X}^{(l)})$ from the $(l+1)$-th layer's graph $\hat{G}^{(l+1)}(\hat{A}^{(l+1)}, \hat{X}^{(l+1)})$. Here, $\hat{A}$ and $\hat{X}$ represent the reconstructed adjacency and node feature matrices. To maintain a consistent layer index, the layer indices in the decoder decrease sequentially. Similar to the encoder, each layer consists of two steps: **Assignment-D** and **Reconstruction**.

Assignment-D: A GNN layer GNN_{r_assign} is used to compute the reconstructed assignment matrix $\hat{S}^{(l)} \in \mathbf{R}^{n_l \times n_{l+1}}$ as

$$\hat{S}^{(l)} = softmax\left(GNN_{r\_assign}\left(\hat{A}^{(l+1)}, \hat{X}^{(l+1)}\right)\right), \quad (5)$$

Reconstruction: We compute the reconstructed node embedding matrix $\hat{Z}^{(l)} \in \mathbf{R}^{n_l \times d_{l+1}}$ as

$$\hat{Z}^{(l)} = GNN_{r\_embed}\left(\hat{A}^{(l+1)}, \hat{X}^{(l+1)}\right). \quad (6)$$

Based on $\hat{Z}^{(l)}$, we perform the matrix computations to obtain the reconstructed embedding graph $\hat{G}^{(l)}(\hat{A}^{(l)}, \hat{X}^{(l)})$ as

$$\hat{A}^{(l)} = S^{(l)} \hat{A}^{(l+1)\top} S^{(l)}, \text{ and } \hat{X}^{(l)} = S^{(l)\top} \hat{Z}^{(l)}. \quad (7)$$

After K decoding layers, we obtain the resulting reconstructed graph of the original sample graph $\hat{G}^{(0)}(\hat{A}^{(0)}, \hat{X}^{(0)})$ that is first input to the encoder of the HA-GAE.

The Loss Function of the HA-GAE

We train the HA-GAE model by minimizing its reconstruction error, and the loss function is defined as

$$L = MSE(A^{(0)}, \hat{A}^{(0)}) + MSE(X^{(0)}, \hat{X}^{(0)}), \quad (8)$$

Here, MSE denotes the mean squared error function. Since we cannot directly generate binary values using graph convolution operations, we use this loss function to ensure that the reconstructed graph structures closely approximate the structures of the original graphs.

Remark 2: Each layer of the HA-GAE model uses the same node assignment information computed by Eq.(2) and Eq.(5). Therefore, the nodes at the same layer of the embedding graphs are transitive-aligned to each other, meaning that the j-th node in graph G naturally corresponds to the j-th node in graph G'. This alignment naturally facilitates the definition of the transitive-aligned graph kernels by measuring the similarity between graphs.

3.4 Construction of the DHTAGK Kernels

In this subsection, we define the proposed DHTAGK kernels. During the HA-GAE training, for each original graph G, we hierarchically generate a series of transitive-aligned embedding graphs $\{G^{(l)}(A^{(l)}, X^{(l)}), l \in \{0, ..., K-1\}\}$ in the encoder. For a pair of original graphs G and G' with different numbers of nodes, their embedding graphs $G^{(l)}$ and $G'^{(l)}$ not only have the same number of nodes but are also transitive-aligned in terms of node correspondences. Using the adjacency and node feature matrices of these embedding graphs, we define three DHTAGK kernels.

The DHTAGK based on Adjacency Matrices (GK-A)

Inspired by [Bai *et al.*, 2015b], we define the GK-A kernel by measuring the similarity between the adjacency matrices of the embedding graphs using the Jensen-Shannon Divergence (JSD). Specifically, for each l-th layer embedding graph $G^{(l)}$, we compute the node probabilities as

$$P^{(l)}(v_i) = \frac{d^{(l)}(v_i)}{\sum_{v_j \in V^{(l)}} d^{(l)}(v_j)}, \tag{9}$$

Here, $P^{(l)}(v_i)$ represents the probability of node v_i at layer l, and $d^{(l)}(v_i)$ represents the degree of node v_i at layer l. For graphs $G^{(l)}$ and $G'^{(l)}$, the GK-A kernel k_A is defined as

$$k_A(G, G') = \sum_l k_A^{(l)}\left(G^{(l)}, G'^{(l)}\right), \tag{10}$$

where each $k_A^{(l)}\left(G^{(l)}, G'^{(l)}\right)$ is the kernel measure between their l-layer embedding graphs $G^{(l)}$ and $G'^{(l)}$ defined as

$$k_A^{(l)}\left(G^{(l)}, G'^{(l)}\right) = e^{-D_{JS}^{(l)}\left(G^{(l)}, G'^{(l)}\right)}. \tag{11}$$

Here $D_{JS}^{(l)}\left(G^{(l)}, G'^{(l)}\right)$ is the JSD measure defined as

$$D_{JS}^{(l)}\left(G^{(l)}, G'^{(l)}\right) = H\left(\frac{P^{(l)}+P'^{(l)}}{2}\right) - \frac{H\left(P^{(l)}\right)+H\left(P'^{(l)}\right)}{2} \tag{12}$$

where $H(\cdot)$ represents Shannon entropy associated with the node distribution $P^{(l)}(v_i)$.

Remark 3: Unlike other similarity measures such as Euclidean distance and cosine similarity, the JSD offers an information-theoretic perspective by quantifying the probability of Steady-State Random Walks visiting nodes, thus capturing the structural information content of graphs. Since the alignment property allows the node indices to naturally correspond to each other, we can directly use JSD to measure the differences between the probabilities of pairwise graphs, effectively capturing structural discrepancies between graphs.

The DHTAGK based on Node Feature Matrices (GK-X)

We define the GK-X kernel by measuring the Gaussian kernel-based similarity between the node feature matrices of the embedding graphs. For the pair of sample graphs G and G', the GK-X kernel k_X is defined as

$$k_X(G, G') = \sum_l k_X^{(l)}\left(G^{(l)}, G'^{(l)}\right). \tag{13}$$

where each $k_X^{(l)}(G^{(l)}, G'^{(l)})$ is the Gaussian kernel between the node feature matrices $X^{(l)}$ and $X'^{(l)}$, which is defined as

$$k_X^{(l)}\left(G^{(l)}, G'^{(l)}\right) = e^{-\frac{\|X^{(l)}-X'^{(l)}\|^2}{2\sigma^2}}. \tag{14}$$

Here, $\|X^{(l)} - X'^{(l)}\|$ is the Euclidean distance between the node feature matrices $X^{(l)}$ and $X'^{(l)}$. σ is a parameter to control the width of the Gaussian graph kernel.

The Composite DHTAGK (GK-C)

We define the GK-C kernel by summing the kernels GK-A and GK-X. For the pair of sample graphs G and G', the GK-C kernel k_C is defined as

$$k_C(G, G') = k_A(G, G') + k_X(G, G'). \tag{15}$$

Clearly, unlike the proposed GK-A and GK-X kernels, the GK-C kernel can simultaneously capture the structural information through either the adjacency matrix or the node feature matrix, preserving richer structural characteristics.

Remark 4: As stated in **Remark 2**, for any pair of sample graphs, the adjacency matrices and the node feature matrices of their embedding graphs extracted from the HA-GAE are transitively aligned. Since the exponentiation of the negative JSD-based measure for the GK-A kernel as well as the Gaussian kernel-based measure for the GK-X kernel are both positive definite (pd) measures, the resulting GK-A and GK-X kernels based on the transitive aligned structure representations are naturally pd. Thus, the GK-C kernel, as the sum of *pd* kernels, is also *pd*.

3.5 Properties of the DHTAGK Kernels

In this subsection, we discuss the theoretical properties of the proposed DHTAGK kernels, explaining the effectiveness.

First, unlike the R-convolution kernels, which focus on small substructures, the proposed DHTAGK kernels are defined based on embedding graphs that progressively shrink in size, from the larger original global graph to the smaller coarsened embedding graph. This gradual reduction in the size of the embedding graphs helps us abstract graph information layer by layer, enabling the capture of both global and local structural information simultaneously. Since the HA-GAE can use all sample graphs for training, the DHTAGK kernels can capture the common structural patterns shared across all graphs. In contrast, the R-convolution kernels only capture information between individual graph pairs.

Second, unlike the R-convolution graph kernels that overlook the structural correspondence information between graphs, the proposed DHTAGK kernels are computed by measuring the similarity between the aligned adjacency matrices and the aligned node feature matrices of the embedding graphs that are constructed by the HA-GAE through the node assignment operation. Thus, the proposed DHTAGK kernels can naturally encapsulate the structural correspondence information through the aligned structure matrices.

Third, most existing matching-based graph kernels suffer from the non-transitive alignment relationships and cannot guarantee the positive definiteness. By contrast, the associated HA-GAE utilizes the same node alignment function to identify the correspondence between nodes of all graphs,

Datasets	MUTAG	PTC_MR	ENZYMES	PROTEINS	D&D	IMDB_B	IMDB_M	REDDIT-B	REDDIT-M
Avg.Nodes	17.9	25.5	32.63	39.1	284.32	19.77	13	429.6	508.8
Graphs	188	344	600	1113	1178	1000	1500	2000	4999
Classes	2	2	6	2	2	2	3	2	5
Domain	Bio	Bio	Bio	Bio	Bio	SN	SN	SN	SN

Table 1: Information of the graph datasets.

and only the nodes assigned to the same cluster can be considered aligned. Thus, the DHTAGK kernels defined based on the transitive-aligned embedding graphs of HA-GAE are transitive-aligned and naturally ***pd***.

4 Experiments

In this section, we evaluate the proposed DHTAGK kernels on eight benchmark graph classification datasets from the domains of bioinformatics (Bio) and social networks (SN). Details of these datasets are shown in Table 1.

Experimental Settings[1]: We set the number of layers for both the HA-GAE encoder and decoder to 3-5. In each layer, the number of nodes in the embedding graph decreases layer by layer according to an assignment ratio 0.5. The feature dimension of the embedding graph is fixed at 32 dimensions. After training the HA-GAE, we input the graph data into the trained HA-GAE to obtain hierarchically aligned embedding graphs, and then compute the GK-A, GK-X, and GK-C graph kernel matrices. Finally, we use the C-SVM implemented by LIBSVM for 10-fold cross-validation and calculate the classification accuracy. We repeat the experiment 10 times and report the average classification accuracy and standard error.

To evaluate the capability on graph classification tasks, we compare our graph kernels with several advanced graph kernels and GNNs. Specifically, **the graph kernels include**: 1) the RWGK [Gartner *et al.*, 2003], 2) GK [Shervashidze *et al.*, 2009], 3) WLSK [Shervashidze *et al.*, 2011], 4) JTQK [Bai *et al.*, 2014] (with $q = 2$), 5) ASK [Bai *et al.*, 2015a], 6) CORE WL & CORE SP [Nikolentzos *et al.*, 2018], 7) EDBMK [Xu *et al.*, 2021], and 8) QBMK [Bai *et al.*, 2024]. **The deep learning methods include** five baselines: 1) the DGCNN [Zhang *et al.*, 2018], 2) DiffPool [Ying *et al.*, 2018], 3) ECC [Simonovsky and Komodakis, 2017], 4) GIN [Xu *et al.*, 2019], and 5) GraphSAGE [Hamilton *et al.*, 2017]; and six advanced GNNs: 1) the DGK [Yanardag and Vishwanathan, 2015], 2) p-RWNN [Nikolentzos and Vazirgiannis, 2020] (with $p = 1, 2, 3$), and 3) GKNN WL & GKNN GL [Cosmo *et al.*, 2024].

The classification accuracy and standard error for each graph kernel method are reported in Table 2, since the alternative kernels are evaluated with the same setup, we directly use the accuracies from the corresponding literature. The results for each deep learning method are shown in Table 3. For the baseline deep learning methods, we adapt the results from the fair comparison [Errica *et al.*, 2020]. For other GNNs, we report the best results from original papers. The *Rank* column in Table 2 and Table 3 shows the mean rank of each method.

[1]Code is available at our GitHub repository: https://github.com/Xiaoqin0421/DHTAGK

Experimental Analysis: The experimental results show that the DHTAGK kernels outperform most graph kernel methods and achieve classification performance comparable to deep learning approaches. While the WLSK and JTQK kernels, as well as deep learning methods, utilize node-label information during training, our method adopts a self-supervised learning strategy for training the HA-GAE. Despite using a shallow learning framework via the C-SVM, the DHTAGK kernels achieve superior performance across most datasets compared to the end-to-end models. This indicates that the autoencoding process effectively captures more informative topological features compared to other approaches. The effectiveness of the DHTAGK kernels is fourfold.

First, unlike the traditional R-convolution kernels, the DHTAGK kernels leverage a hierarchical approach to simultaneously capture both local and global structural information, resulting in superior expressive power. **Second**, unlike the traditional R-convolution kernels and deep learning approaches, the DHTAGK kernels ensure precise node correspondences at various levels through their hierarchical alignment mechanism. This allows for a more comprehensive capture of the structural features. In contrast, other models often struggle to accurately identify substructure correspondences within complex graphs, leading to suboptimal performance. **Third**, compared to the ASK kernel, the correspondence information identified by the DHTAGK kernels is transitive. By contrast, the ASK kernel cannot guarantee the transitivity. **Fourth**, the DHTAGK kernels excel in extracting shared structural information across graph samples. Unlike other classical graph kernel methods, which typically focus on pairwise similarities, our kernels aggregate information from all samples, capturing a broader range of structural patterns. This comprehensive feature extraction results in a more detailed representation of the graph structure, making the DHTAGK particularly effective for complex datasets.

Ablation Study. We compare the performance of the GK-A, GK-X, and GK-C, and attribute the differences to the distinct types of graph information they capture. Specifically, the GK-A emphasizes global structural information through adjacency matrices, yielding better results on datasets with prominent structural characteristics, such as the MUTAG. However, the GK-A does not consistently achieve optimal performance, as it fails to incorporate node attribute information. In contrast, the GK-X focuses on node features and performs better on datasets rich in attribute information, such as the PTC_MR. Finally, the GK-C integrates both structural and feature information, enabling it to capture graph characteristics more comprehensively.

Hyperparameter Analysis. We investigate the impact of the number of auto-encoder layers and the embedding graph

Model	MUTAG	PTC_MR	ENZYMES	PROTEINS	D&D	IMDB_B	IMDB_M	REDDIT-B	Rank
GK-A	**88.72±0.39**(1)	58.29±0.71(6)	42.43±0.22(6)	**75.18±0.31**(2)	78.65±0.27(6)	73.19±0.23(4)	50.20±0.20(5)	89.17±0.17(4)	4.25
GK-X	88.50±0.85(3)	**61.24±0.50**(1)	55.88±0.45(3)	**75.42±0.41**(1)	79.16±0.41(5)	**74.1±0.37**(1)	50.23±0.27(4)	89.80±0.17(3)	**2.62**
GK-C	87.39±0.69(7)	**60.71±0.28**(2)	**56.68±0.65**(1)	**74.95±0.31**(3)	**80.03±0.26**(1)	73.37±0.33(3)	50.29±0.22(3)	89.91±0.15(2)	**2.87**
RWGK	80.77±0.72(12)	55.91±0.37(10)	22.37±0.35(10)	74.20±0.40(4)	71.70±0.47(11)	67.94±0.77(10)	46.72±0.30(9)	72.73±0.39(9)	9.37
GK	81.66±0.11(11)	—	24.87±0.22(9)	71.67±0.55(7)	78.45±0.26(7)	65.87±0.98(9)	45.42±0.87(10)	77.34±0.18(7)	8.57
WLSK	82.88±0.57(10)	56.05±0.51(9)	52.75±0.44(4)	73.52±0.43(5)	79.78±0.36(2)	71.88±0.77(7)	49.50±0.49(7)	76.56±0.30(8)	6.50
JTQK	85.50±0.55(9)	57.39±0.46(7)	56.41±0.42(2)	72.86±0.41(6)	79.49±0.32(3)	72.45±0.81(6)	50.33±0.49(2)	77.60±0.35(6)	5.12
ASK	87.50±0.65(5)	—	—	—	—	70.38±0.72(8)	50.12±0.51(6)	—	6.33
CORE WL	87.47±1.08(6)	59.43±1.20(3)	47.82±4.62(5)	—	79.24±0.34(4)	74.02±0.42(2)	**51.35±0.48**(1)	78.02±0.23(5)	4.00
CORE SP	88.29±1.08(4)	59.06±0.93(5)	41.20±1.21(7)	—	77.30±0.80(10)	72.62±0.59(5)	49.43±0.42(8)	**90.84±0.14**(1)	3.71
EDBMK	86.35(8)	56.75(8)	36.85(8)	—	78.19(8)	—	—	—	8.00
QBMK	88.55±0.43(2)	59.38±0.36(4)	—	—	77.60±0.47(9)	—	—	—	5.00

Table 2: Classification accuracy (in % ± standard error) for comparisons with graph kernels.

	MUTAG	PTC_MR	ENZYMES	PROTEINS	D&D	IMDB_B	IMDB_M	REDDIT-B	REDDIT-M	Rank
GK-A	88.72±0.39(2)	58.29±0.71(5)	42.43±0.22(10)	75.18±0.31(4)	**78.65±0.27**(3)	**73.19±0.23**(3)	**50.20±0.20**(3)	89.17±0.17(7)	53.16±0.11(4)	**4.55**
GK-X	88.50±0.85(4)	**61.24±0.50**(1)	55.88±0.45(8)	75.42±0.41(2)	**79.16±0.41**(2)	**74.1±0.37**(1)	**50.23±0.27**(2)	89.80±0.17(5)	50.76±0.17(7)	**3.55**
GK-C	87.39±0.69(6)	**60.71±0.28**(2)	56.68±0.65(7)	74.95±0.31(5)	**80.03±0.26**(1)	**73.37±0.33**(2)	**50.29±0.22**(1)	89.91±0.15(3)	50.19±0.17(8)	**3.88**
DGCNN	—	—	38.9±5.7(11)	72.9±3.5(12)	76.6±4.3(8)	69.2±3.0(10)	45.6±3.4(10)	87.8±2.5(9)	49.2±1.2(10)	10.00
DiffPool	—	—	59.5±5.6(2)	73.7±3.5(9)	75.0±3.5(10)	68.4±3.3(12)	45.6±3.4(10)	89.1±1.6(8)	53.8±1.4(2)	7.57
ECC	—	—	29.5±8.2(12)	72.3±3.4(13)	72.6±4.1(12)	67.7±2.8(13)	43.5±3.1(12)	—	—	12.40
GIN	—	—	**59.6±4.5**(1)	73.3±4.0(10)	75.3±2.9(9)	71.2±3.9(4)	48.5±3.3(5)	89.9±1.9(4)	**56.1±1.7**(1)	4.85
GraphSAGE	—	—	58.2±6.0(3)	73.0±4.5(11)	72.9±2.0(11)	68.8±4.5(11)	47.6±3.5(8)	84.3±1.9(10)	50.0±1.3(9)	9.00
DGK	82.66±1.45(9)	57.32±1.13(6)	53.4±0.9(9)	71.68±0.50(14)	78.50±0.22(4)	66.96±0.56(14)	44.55±0.52(11)	77.34±0.18(11)	41.27±0.18(13)	10.11
1-RWNN	**89.2±4.3**(1)	—	56.7±5.2(6)	**75.7±3.3**(1)	77.6±4.7(5)	70.8±4.8(6)	47.8±3.8(7)	**90.4±1.9**(1)	51.7±1.5(5)	4.00
2-RWNN	88.1±4.8(5)	—	57.4±4.9(5)	74.1±3.3(8)	76.9±4.6(7)	70.6±4.4(7)	48.8±2.9(4)	90.3±1.8(2)	51.7±1.4(6)	5.50
3-RWNN	88.6±4.1(3)	—	57.6±6.3(4)	74.3±3.3(7)	77.4±4.9(6)	70.7±3.9(5)	47.8±3.5(7)	89.7±1.2(6)	53.4±1.6(3)	5.12
GKNN WL	85.73±2.70(7)	59.29±2.54(4)	—	74.94±1.10(6)	—	69.70±2.20(9)	47.87±1.78(6)	—	47.87±1.78(11)	7.12
GKNN GL	85.24±2.28(8)	60.13±1.94(3)	—	75.36±1.12(3)	—	69.90±1.44(8)	45.67±1.22(9)	—	45.67±1.22(12)	7.17

Table 3: Classification accuracy (in % ± standard error) for comparisons with deep learning methods.

assignment ratio using the PROTEINS dataset. Figure 4 and Figure 5 show the classification accuracy of the GK-A, GK-X, and GK-C at different layers and assignment ratios.

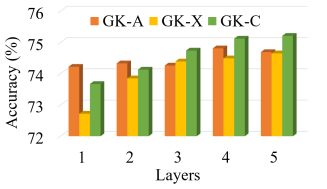

Figure 4: Performance vs. Layers

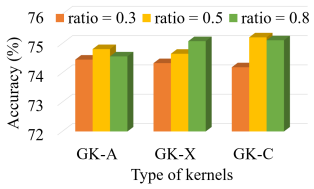

Figure 5: Performance vs. Ratio.

Impact of the Layers. As the number of layers increases, the classification accuracy of the GK-A, GK-X, and GK-C improves. The GK-A performs better with fewer layers, due to its ability to capture local node and neighborhood information at shallow levels. When the model reaches three layers, the GK-C outperforms the GK-A, as it offers a more comprehensive representation in both structure and features.

Impact of the Ratio. It can be seen in Figure 5 that an assignment ratio of 0.5 ensures that the size of the embedding graph is moderate, balancing local and global information. Specifically, a smaller assignment ratio leads to small embedding graphs, causing over-squashing and information loss. A larger assignment ratio may cause the graph kernel to focus more on large-scale structures, potentially neglecting fine-grained local information, and the slower reduction in the size of the embedding graph decreases computational efficiency. Therefore, choosing a moderate assignment ratio of 0.5 achieves a good balance between local and global information, and enhances computational efficiency.

Visualization. Figure 6 shows several molecular graphs from the MUTAG dataset, where nodes with the same color are aligned to the same cluster. The experimental results demonstrate that the HA-GAE effectively captures meaningful node correspondence information, such as ring structures and functional groups commonly found in molecular graphs.

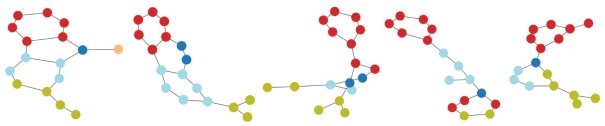

Figure 6: A few visualizations showcasing node alignment results.

5 Conclusion

In this paper, we have proposed a novel family of DHTAGK kernels for graph classification. The DHTAGK kernels are defined based on the transitive-aligned adjacency and node feature matrices of the hierarchical embedding graphs extracted from the newly developed HA-GAE model. These kernels not only overcome the issue of ignoring structural correspondences between substructures in the existing R-convolution kernels, but also address the non-positive definiteness problem in node-based alignment kernels. Furthermore, we reflect both local and global graph characteristics, and capture the structural patterns across all graphs, through the HA-GAE. Experiments demonstrate the effectiveness of the proposed kernels. In future work, we plan to explore additional forms of transitive alignment information produced by the HA-GAE to design new graph kernels. We also intend to develop a mechanism for adaptively computing the weights of the GK-A and GK-X kernels.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants T2122020, 61602535, U21A20473, and 62172370. This work is also supported in part by the Humanity and Social Science Foundation of Ministry of Education (24YJAZH022), and the Program for Innovation Research in the Central University of Finance and Economics.

References

[Aziz *et al.*, 2013] Furqan Aziz, Richard C. Wilson, and Edwin R. Hancock. Backtrackless Walks on a Graph. *TNNLS*, pages 977–989, 2013.

[Bai *et al.*, 2014] Lu Bai, Luca Rossi, Horst Bunke, and Edwin R. Hancock. Attributed Graph Kernels Using the Jensen-Tsallis q-Differences. In *Proceedings of ECML-PKDD*, pages 99–114, 2014.

[Bai *et al.*, 2015a] Lu Bai, Luca Rossi, Zhihong Zhang, and Edwin R. Hancock. An Aligned Subtree Kernel for Weighted Graphs. In *Proceedings of ICML*, pages 30–39, 2015.

[Bai *et al.*, 2015b] Lu Bai, Zhihong Zhang, Chaoyan Wang, Xiao Bai, and Edwin R. Hancock. A Graph Kernel Based on the Jensen-Shannon Representation Alignment. In *Proceedings of IJCAI*, pages 3322–3328, 2015.

[Bai *et al.*, 2019] Lu Bai, Yuhang Jiao, Lixin Cui, and Edwin R. Hancock. Learning Aligned-Spatial Graph Convolutional Networks for Graph Classification. In *Proceedings of ECML-PKDD*, pages 464–482, 2019.

[Bai *et al.*, 2020] Lu Bai, Luca Rossi, Lixin Cui, Jian Cheng, and Edwin R. Hancock. A Quantum-inspired Similarity Measure for the Analysis of Complete Weighted Graphs. *IEEE Trans. Cybern.*, pages 1264–1277, 2020.

[Bai *et al.*, 2022] Lu Bai, Lixin Cui, and Edwin R. Hancock. A Hierarchical Transitive-Aligned Graph Kernel for Un-Attributed Graphs. In *Proceedings of ICML*, pages 1327–1336, 2022.

[Bai *et al.*, 2024] Lu Bai, Lixin Cui, Ming Li, Yue Wang, and Edwin Hancock. QBMK: Quantum-based Matching Kernels for Un-attributed Graphs. In *Proceedings of ICML*, pages 2364–2374, 2024.

[Borgwardt and Kriegel, 2005] K.M. Borgwardt and H. Kriegel. Shortest-path Kernels on Graphs. In *Proceedings of ICDM*, pages 74–81, 2005.

[Cosmo *et al.*, 2024] Luca Cosmo, Giorgia Minello, Alessandro Bicciato, Michael M. Bronstein, Emanuele Rodolà, Luca Rossi, and Andrea Torsello. Graph Kernel Neural Networks. *TNNLS*, pages 1–14, 2024.

[Cui *et al.*, 2024] Lixin Cui, Lu Bai, Xiao Bai, Yue Wang, and Edwin R. Hancock. Learning Aligned Vertex Convolutional Networks for Graph Classification. *TNNLS*, pages 4423–4437, 2024.

[Errica *et al.*, 2020] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A Fair Comparison of Graph Neural Networks for Graph Classification. In *Proceedings of ICLR*, 2020.

[Gartner *et al.*, 2003] Thomas Gartner, Peter A. Flach, and Stefan Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Proceedings of COLT*, pages 129–143, 2003.

[Gkirtzou and Blaschko, 2016] Katerina Gkirtzou and Matthew B. Blaschko. The Pyramid Quantized Weisfeiler-Lehman Graph Representation. *Neural Comput.*, pages 1495–1507, 2016.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Proceedings of NeurIPS*, pages 2672–2680, 2014.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Proceedings of NeurIPS*, pages 1025–1035, 2017.

[Harchaoui and Bach, 2007] Zaid Harchaoui and Francis Bach. Image Classification with Segmentation Graph Kernels. In *Proceedings of CVPR*, pages 1–8, 2007.

[Haussler, 1999] D. Haussler. Convolution Kernels on Discrete Structures. 1999.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, pages 1735–1780, 1997.

[Johansson *et al.*, 2014] Fredrik D. Johansson, Vinay Jethava, Devdatt Dubhashi, and Chiranjib Bhattacharyya. Global Graph Kernels using Geometric Embeddings. In *Proceedings of ICML*, pages 694–702, 2014.

[Kashima *et al.*, 2003] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized Kernels Between Labeled Graphs. In *Proceedings of ICML*, pages 321–328, 2003.

[Kipf and Welling, 2016] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR*, 2017.

[Nikolentzos and Vazirgiannis, 2020] Giannis Nikolentzos and Michalis Vazirgiannis. Random Walk Graph Neural Networks. In *Proceedings of NeurIPS*, pages 16211–16222, 2020.

[Nikolentzos *et al.*, 2018] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. A Degeneracy Framework for Graph Similarity. In *Proceedings of IJCAI*, pages 2595–2601, 2018.

[Pan *et al.*, 2020] Shirui Pan, Ruiqi Hu, Sai-Fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning Graph Embedding with Adversarial Training Methods. *IEEE Trans. Cybern.*, pages 2475–2487, 2020.

[Shervashidze *et al.*, 2009] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten

Borgwardt. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of AISTATS*, pages 488–495, 2009.

[Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-Lehman Graph Kernels. *JMLR*, pages 2539–2561, 2011.

[Simonovsky and Komodakis, 2017] Martin Simonovsky and Nikos Komodakis. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In *Proceedings of CVPR*, pages 29–38, 2017.

[Xu *et al.*, 2018] Lixiang Xu, Xiaoyi Jiang, Lu Bai, Jin Xiao, and Bin Luo. A Hybrid Reproducing Graph Kernel based on Information Entropy. *PR*, pages 89–98, 2018.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *Proceedings of ICLR*, 2019.

[Xu *et al.*, 2021] Lixiang Xu, Lu Bai, Xiaoyi Jiang, Ming Tan, Daoqiang Zhang, and Bin Luo. Deep Rényi Entropy Graph kernel. *PR*, 2021.

[Yanardag and Vishwanathan, 2015] Pinar Yanardag and S.V.N. Vishwanathan. Deep Graph Kernels. In *Proceedings of KDD*, pages 1365–1374, 2015.

[Ying *et al.*, 2018] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proceedings of NeurIPS*, page 4805–4815, 2018.

[Yu *et al.*, 2018] Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C. Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning Deep Network Representations with Adversarially Regularized Autoencoders. In *Proceedings of KDD*, pages 2663–2671, 2018.

[Zhang *et al.*, 2018] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An End-to-End Deep Learning Architecture for Graph Classification. In *Proceedings of AAAI*, pages 4438–4445, 2018.