Deep Hierarchical Graph Alignment Kernels

Shuhao Tang[1] , **Hao Tian**[1] , **Xiaofeng Cao**[2] and **Wei Ye**[1,*]
[1]Tongji University, Shanghai, China
[2]Jilin University, Changchun, China
{tangsh2022, yew}@tongji.edu.cn
xiaofengcao@jlu.edu.cn

Abstract

Typical $\mathcal{R}$-convolution graph kernels invoke the kernel functions that decompose graphs into non-isomorphic substructures and compare them. However, overlooking implicit similarities and topological position information between those substructures limits their performances. In this paper, we introduce Deep Hierarchical Graph Alignment Kernels (DHGAK) to resolve this problem. Specifically, the relational substructures are hierarchically aligned to cluster distributions in their deep embedding space. The substructures belonging to the same cluster are assigned the same feature map in the Reproducing Kernel Hilbert Space (RKHS), where graph feature maps are derived by kernel mean embedding. Theoretical analysis guarantees that DHGAK is positive semi-definite and has linear separability in the RKHS. Comparison with state-of-the-art graph kernels on various benchmark datasets demonstrates the effectiveness and efficiency of DHGAK. The code is available at Github (https://github.com/EWesternRa/DHGAK).

1 Introduction

Over the past decades, more and more data in studies depicting relationships between objects cannot be simply interpreted as vectors or tables. Instead, they are naturally represented by graphs. In order to capture the inherent information from graph-structured data, many graph-based machine learning algorithms are gaining attraction.

Graph kernels (GKs) [Vishwanathan *et al.*, 2010] are conventional methods for measuring the similarity of graphs by kernel method defined on graphs. Most graph kernels are instances of $\mathcal{R}$-convolution theory [Haussler and others, 1999]. These methods decompose graphs into non-isomorphic substructures (e.g., paths or walks, subgraphs, subtrees) and compare them. Graph similarity is the sum of all the substructure similarities. However, the computation of the substructure similarity is strict, i.e., all non-isomorphic substructures are treated as dissimilar ones, which impedes the precise comparison of graphs. In addition, $\mathcal{R}$-convolution graph kernels neglect the topological position information between substructures, which plays a crucial role in the characteristic of the whole graph, e.g., whether or not a drug molecule is effective on a disease.

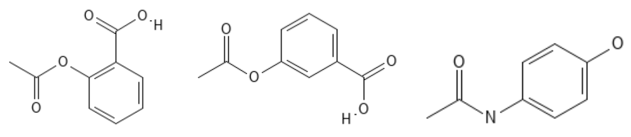

(a) Aspirin (b) 3-Acetoxybenzoic acid (c) Acetaminophen

Figure 1: The structure 2D depictions of Aspirin, Acetaminophen, and 3-Acetyloxybenzoic acid.

We show the 2D representations of three molecules in Figure 1 to explain the importance of precise substructure similarity comparison and their relative topological positions. Aspirin (Figure 1 (a)) and 3-Acetoxybenzoic acid (Figure 1 (b)) have the same benzoic and acetyloxy substructures, but the former is commonly used in drugs for the treatment of pain and fever, while the latter is not. The difference in the relative topological positions of these two substructures causes the different molecule characteristics. $\mathcal{R}$-convolution graph kernels are difficult to distinguish the two molecules because they are unaware of the substructure topological position information. Aspirin and Acetaminophen (Figure 1 (c)) have similar substructures acetylamino and acetyloxy (only a little bit of difference in the atoms), they are both used for pain and fever. However, $\mathcal{R}$-convolution graph kernels cannot precisely compute the similarity between these similar substructures from them.

Many graph kernels try to alleviate these problems in different ways. WWL [Togninalli *et al.*, 2019] embeds graph substructures (WL [Shervashidze *et al.*, 2011] subtree patterns) into continuous vectors and compares their distributions by leveraging the Wasserstein distance. The use of optimal transport in WWL implicitly aligns WL subtree patterns from different graphs, thus addressing the neglect of the topological position information between WL subtree patterns. But WWL has high computational overhead and is not positive semi-definite, since the optimal transport is not guaranteed to satisfy the transitivity. In other words, if σ is the alignment between substructures from graphs G_1 and

[*] Corresponding author.

G_2 and π is the alignment between substructures from graphs G_2 and G_3, the optimal transport cannot guarantee that the alignment between substructures from graphs G_1 and G_3 is $\sigma \circ \pi$. GAWL [Nikolentzos and Vazirgiannis, 2023] is a positive semi-definite graph alignment kernel that explicitly aligns WL subtree patterns by comparing their labels refined by the Weisfeiler-Leman graph isomorphism test [Leman and Weisfeiler, 1968]. However, with more iterations of the WL-relabeling process, it is more difficult to align since all the WL subtree patterns tend to have different labels.

To mitigate the above problems, we propose a new graph kernel called Deep Hierarchical Graph Alignment Kernels (DHGAK). Specifically, we define a new graph substructure named b-width h-hop slice around a node to capture the structural information. Then, we embed each slice into a deep vector space by Natural Language Models (NLM) for precise comparison. In the deep embedding space, we propose a Deep Alignment Kernel (DAK) that aligns slices from the same or different graphs by computing the expectation of the cluster assignments of the adopted clustering methods. Slices from the same cluster are aligned while those from different clusters are unaligned. We prove that the alignment by clustering in DAK satisfies the transitivity. By assuming the feature maps of DAK in one graph are i.i.d. (independent and identically distributed) samples of an unknown distribution, we adopt the kernel mean embedding [Muandet *et al.*, 2017] to construct Deep Graph Alignment Kernel (DGAK) from a distributional perspective. With the summation of DGAK on each hop of b-width slice, we exploit hierarchical graph structural information and finally get our Deep Hierarchical Graph Alignment Kernels (DHGAK). All the three kernels, i.e., DAK, DGAK, and DHGAK, are positive semi-definite. It is theoretically demonstrated that there exists a family of clustering methods derived from which the feature map of our proposed DHGAK is linearly separable for any dataset in the Reproducing Kernel Hilbert Space (RKHS).

Our contributions are summarized as follows:

- We propose a generalized graph alignment kernel DHGAK that can use any NLM to generate deep embeddings of substructures and any clustering methods to align substructures. DHGAK can more precisely compute graph similarity compared to conventional $\mathcal{R}$-convolution graph kernels by not only taking into account the distributions of substructures but also their topological position information in graphs.
- We prove that DHGAK is positive semi-definite and satisfies alignment transitivity. Theoretical analysis proves that the feature map of DHGAK produced by certain clustering methods is linearly separable in the RKHS.
- We compare DHGAK with state-of-the-art graph kernels on various benchmark datasets and the experimental results show that DHGAK has the best classification accuracy on most datasets.

2 Related Work

Weisfeiler-Lehman graph kernel (WL) [Shervashidze *et al.*, 2011] quantifies graph similarity by calculating the number of common labels (subtree patterns) between two graphs at each iteration of the WL-relabeling process (Weisfeiler-Lehman graph isomorphism test [Leman and Weisfeiler, 1968]), which however diverges too fast and leads to coarse graph similarity. Wasserstein Weisfeiler-Lehman graph kernel (WWL) [Togninalli *et al.*, 2019] extends WL to graphs with continuous node attributes. It uses Wasserstein distance to measure the distance between graphs but the cost of computation is very expensive. Gradual Weisfeiler-Lehman (GWL) [Bause and Kriege, 2022] introduces a framework that allows a slower divergence of the WL relabeling process. GAWL [Nikolentzos and Vazirgiannis, 2023] initiates the alignment of nodes across distinct graphs based on their shared labels, determined by the WL-relabeling process. It becomes difficult for node alignment if the iteration number of the WL-relabeling process increases.

Shortest Path graph kernels (SP) [Borgwardt and Kriegel, 2005] quantify graph similarity by evaluating the shortest paths that have the same labels of the source and sink nodes and the same length between all node pairs within two graphs. However, the shortest path representation is too simple and may lose information. Tree++ [Ye *et al.*, 2019] and MWSP [Ye *et al.*, 2023] extract the multi-scale shortest-path features in the BFS tree rooted at each node of graphs. RetGK [Zhang *et al.*, 2018] represents each graph as a set of multi-dimensional vectors consisting of return probability resulting from random walks starting and ending at identical nodes. The embedding in the RKHS is achieved via Maximum Mean Discrepancy (MMD) [Gretton *et al.*, 2012].

Beyond the graph kernels mentioned above, assignment kernels have attracted the attention of scholars. The main idea is to compute an optimal matching between the substructures of each graph pair, reflecting the graph similarity. WL-OA [Kriege *et al.*, 2016] first designs some base kernels to generate hierarchies and then computes the optimal assignment kernels. Such kernels are guaranteed to be positive semi-definite. PM [Nikolentzos *et al.*, 2017] adopts the Pyramid Match kernel [Lazebnik *et al.*, 2006] to find an approximate correspondence between two graph embeddings (the eigenvectors of graph adjacency matrix). HTAK [Bai *et al.*, 2022] computes the H-hierarchical prototype representations and hierarchically aligns the nodes of each graph to their different h-level ($1 \leq h \leq H$) prototypes. However, HTAK can only work for un-attributed graphs.

Other graph kernels are as follows. Graph Neural Tangent Kernels (GNTKs) [Du *et al.*, 2019], which correspond to infinite-width multilayer Graph Neural Networks (GNNs) trained by gradient descent, have the same rich expressiveness as GNNs and inherit the advantages of graph kernels. Graph Quantum Neural Tangent Kernel (GraphQNTK) [Tang and Yan, 2022], a novel quantum graph learning model, is designed to capture structural information and exploit quantum parallelism for computational efficiency. The model approximates the underlying GNTKs by introducing a multi-head quantum attention mechanism. Although both GNTK and GraphQNTK exploit the high expressiveness of GNNs, they suffer from the overparameterization and overfitting issues of GNNs on small graphs.

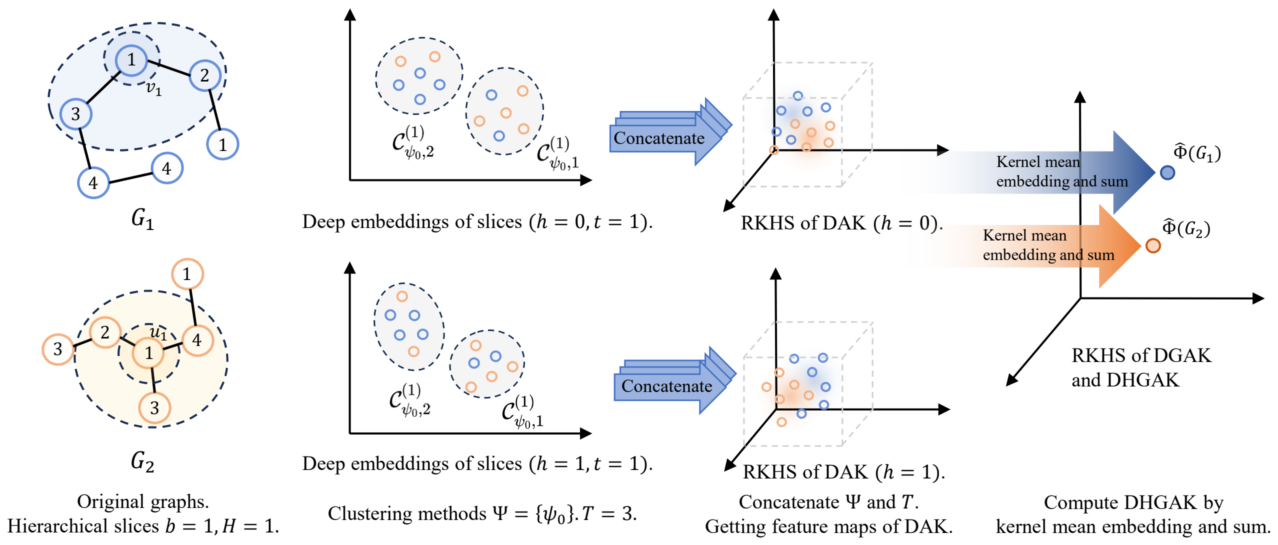

v_1
G_1
u_1
G_2
Original graphs.
Hierarchical slices $b = 1, H = 1$.
$\mathcal{C}^{(1)}_{\psi_0,2}$
$\mathcal{C}^{(1)}_{\psi_0,1}$
Deep embeddings of slices ($h = 0, t = 1$).
Deep embeddings of slices ($h = 1, t = 1$).
Clustering methods $\Psi = \{\psi_0\}. T = 3$.
Concatenate
RKHS of DAK ($h = 0$).
RKHS of DAK ($h = 1$).
Concatenate Ψ and T.
Getting feature maps of DAK.
Kernel mean embedding and sum
$\hat{\Phi}(G_1)$
$\hat{\Phi}(G_2)$
RKHS of DGAK and DHGAK
Compute DHGAK by kernel mean embedding and sum.

Figure 2: The framework of DHGAK is presented with slice width $b = 1$, maximum hop $H = 1$, clustering method set $\Psi = \{\psi_0\}$, and experiment times $T = 3$. For convenience, we illustrate our method from the perspective of space transformation. First, we construct the hierarchical b-width h-hop slice for each node in G_1 and G_2, where $b = 1$ and h is taken from 0 to 1. Next, the encoding method mentioned in Section 3.2 is used to obtain the encoding sequence of slice $S^b_h(v)$. The numbers in nodes represent the node labels in the original graphs, we can get $S^1_0(v_1) = [1, 3, 2]$, $S^1_1(v_1) = [3, 1, 4, 2, 1, 1]$, $S^1_0(u_1) = [1, 2, 4, 3]$, and $S^1_1(u_1) = [2, 1, 3, 4, 1, 1, 3, 1]$. Then, the encoding of the slice is embedded into deep embedding space via a Natural Language Model and updated by Equation 1. Within this deep embedding space, we cluster similar slices and concatenate the cluster indicators for all clustering methods Ψ and experiment times T to obtain the feature maps of DAK. $\mathcal{C}^{(t)}_{\psi_0,i}$ represents the i-th cluster at the t-th experiment under clustering method $\psi_0 \in \Psi$. Finally, the feature map of DGAK is the kernel mean embeddings of the DAK feature maps. The feature map of DHGAK is computed as the sum of those of DGAK on slices of different hierarchies.

3 The Model

DHGAK defines graph kernels by hierarchically aligning relational substructures to cluster distributions. The framework of DHGAK is presented in Figure 2. Specifically, Section 3.1 introduces the notations used in this paper; Section 3.2 describes how to encode a slice into a unique sequence and then embed it into a deep vector space via Natural Language Models (NLM); In Section 3.3, the Deep Alignment Kernel (DAK) is defined and the feature map of DHGAK is obtained in a distribution perspective; Section 3.4 gives the theoretical analysis of DHGAK.

3.1 Notation

The notations used in this paper are as follows. We consider undirected and labeled graphs. Given a graph dataset $\mathcal{D} = \{G_1, ..., G_N\}$, graph $G_i = (\mathcal{V}_i, \mathcal{E}_i, l)$, where $\mathcal{V}_i$ and $\mathcal{E}_i$ are the sets of nodes and edges of graph G_i, respectively. $l : \mathcal{V}_i \to \Sigma$ is a function that assigns labels in the label alphabet Σ to nodes. For the graph classification problem, each graph G_i is assigned a class label $y_i \in \{1, 2, \ldots, c\}$ (c is the number of classes).

3.2 Hierarchical Neighborhood Structure Encoding

First, we define the b-width h-hop slice of a node as follows:

Definition 1 (b-width h-hop Slice of Node). *Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$, rooted at each node $v \in \mathcal{V}$, we build a truncated BFS tree T_v of depth r. All the nodes in T_v denoted as $N^r(v)$ and all the leaf nodes (nodes in the r-th depth) in T_v denoted as $N_r(v)$ are defined as the r-width and the r-hop neighbors of v, respectively. Then, the b-width h-hop slice of v can be defined as $N^b_h(v) = \bigcup_{u \in N_h(v)} N^b(u)$.*

It is worth noting that when $b = 0$ the b-width neighbor of each node $v \in \mathcal{V}$ is the node itself, which is also equivalent to its b-hop neighbor, i.e., $N^0(v) = N_0(v) = \{v\}$. In Figure 3 (a), we set the slice width b as 0 and vary the slice hop h from 0 to 2. Then we get the 0-width 0-hop slice, 0-width 1-hop slice and 0-width 2-hop slice of v_1, i.e., $N^0_0(v_1)$, $N^0_1(v_1)$ and $N^0_2(v_1)$, which capture the neighborhood structure information of v_1 in different hierarchies. In Figure 3 (b), we show that the 1-width 1-hop slice $N^1_1(v_1)$ of node v_1 is the union set of $N^1(v_2)$, $N^1(v_3)$, and $N^1(v_4)$.

Then, we encode each slice into a unique sequence $S^b_h(v)$. Given width b and hop h, for each node v in the graph, we find its h-hop neighbors $N_h(v)$ and sort them by their eigenvector centrality (EC) [Bonacich, 1987] values in descending order. For each node u in the sorted $N_h(v)$, we concatenate the nodes in its i-hop neighbors $N_i(u)$, sorted by EC in descending order to get $S_i(u)$. Then, we concatenate $S_i(u)$ from $i = 0$ to b to obtain $S^b(u)$. The encoding of slice $S^b_h(v)$ is the concatenation of $S^b(u)$ for all the nodes u in $N_h(v)$. Finally, each node in the encoding of the slice is assigned a label by applying $l(\cdot)$. For example, in Figure 3 (a), $S^0_1(v_1) = l([v_4, v_3, v_2])$; and in Figure 3 (b), $S^1_1(v_1) = l([v_4, v_9, v_1, v_{10}, v_5] \;\|\; [v_2, v_1, v_8, v_7] \;\|\; [v_3, v_1, v_5, v_6])$, where $\|$ concatenates different $S^b(u)$. The pseudo-code is

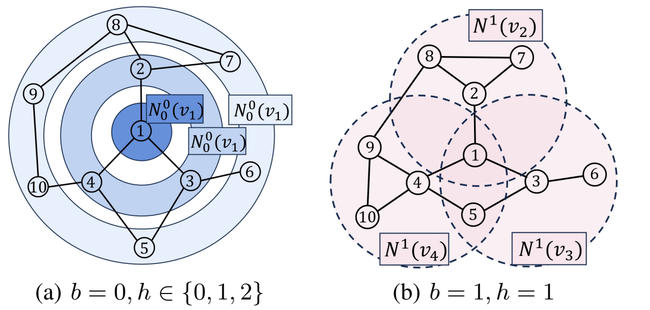


Figure 3: (a) shows b-width h-hop slice of node v_1, where b is fixed as 0 and h ranges from 0 to 2. (b) shows 1-width 1-hop slice of node v_1, where $N_1^1(v_1)$ is the union set of $N^1(v_2), N^1(v_3)$, and $N^1(v_4)$ since $v_2, v_3, v_4 \in N_1(v_1)$.

given in Algorithm 1 in B.1 [1].

In $\mathcal{R}$-convolution graph kernels, the similarity value between each pair of slices is one (if they have the same encoding) or zero (if they have different encodings even the difference is just a little bit). This strict comparison does not consider the fact that there exists some similarity even in two slices that have different encodings. Several studies [Perozzi *et al.*, 2014; Yanardag and Vishwanathan, 2015a] have shown that the substructures in graphs follow a power-law distribution, just like the word distribution in natural language. Thus, to circumvent the above difficulty, we generate a deep embedding for each slice to soften the strict similarity comparison and to compute slice similarity more accurately by employing Natural Language Models (NLM).

More specifically, given a slice encoding, we consider each node label in the encoding as a word and the whole encoding as a sentence. Then an NLM can be trained to learn the embedding of each node label. We define the deep embedding of a slice as follows:

Definition 2 (Deep Slice Embedding)**.** *Given a label embedding function (learned by NLMs) $g : \Sigma \to \mathbb{R}^d$, where d is the dimension of the embedding. The embedding of a b-width h-hop slice of node v is defined as follows:*

$$\mathbf{x}_h^b(v) = \alpha \cdot \mathbf{x}_{h-1}^b(v) + \sum_{u \in S_h^b(v)} g(u) \tag{1}$$

where $\mathbf{x}_0^b(v) := g(l(v))$, α is a decay coefficient of the $(h-1)$-th hop's node label embedding and $S_h^b(v)$ is the encoding of the b-width h-hop slice of node v.

3.3 Deep Hierarchical Graph Alignment Kernels

To consider the topological position information of slices, we propose to align slices from different graphs by clustering in their deep embedding space. Slices within the same cluster are aligned and have the same feature map. This not only addresses the neglect of the topological position information of substructures in conventional $\mathcal{R}$-convolution graph kernels but also reduces the computational overhead since comparison will only be carried out on slices from different clusters. We first define the Deep Alignment Kernel as follows:

[1]All appendices can be found in the Arxiv version of this paper.

Definition 3 (Deep Alignment Kernel (DAK))**.** *Let $\Psi \subset \mathfrak{M}$ be an arbitrary set of clustering methods chosen from all admissible clustering methods $\mathfrak{M}$. Let $U(\Psi)$ denote a uniform distribution on Ψ. For any clustering method $\psi : \mathbf{X}_h^b \to \mathcal{C}_\psi$ in Ψ, it separates each point (b-width h-hop slice embedding) $\mathbf{x} \in \mathbf{X}_h^b$ into one of $|\mathcal{C}_\psi|$ clusters, i.e., $\mathcal{C}_\psi = \{\mathcal{C}_{\psi,1}, \mathcal{C}_{\psi,2}, \ldots, \mathcal{C}_{\psi,|\mathcal{C}_\psi|}\}$. The Deep Alignment Kernel quantifies the likelihood of assigning two points (slice embeddings), $\mathbf{x}, \mathbf{y} \in \mathbf{X}_h^b$, to the same cluster across all clustering methods in the set Ψ, which is defined as follows:*

$$\kappa_h^b(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{\psi \sim U(\Psi)}[\mathbb{I}(\mathbf{x}, \mathbf{y} \in \mathcal{C} | \mathcal{C} \in \mathcal{C}_\psi)] \tag{2}$$

Given that many clustering methods (e.g., K-means [MacQueen and others, 1967]) heavily rely on the initialization points, we perform each clustering method T times to approximate the expectation. The approximation is defined as:

$$\kappa_h^b(\mathbf{x}, \mathbf{y}) \simeq \frac{1}{T|\Psi|} \sum_{\psi \in \Psi} \sum_{t=1}^{T} \sum_{i=1}^{|\mathcal{C}_\psi|} \mathbb{I}(\mathbf{x} \in \mathcal{C}_{\psi,i}^{(t)}) \mathbb{I}(\mathbf{y} \in \mathcal{C}_{\psi,i}^{(t)}) \tag{3}$$

where $\mathbb{I}(a)$ is the indicator function that $\mathbb{I}(a) = 1$ if a is true, otherwise $\mathbb{I}(a) = 0$. $\mathcal{C}_{\psi,i}^{(t)}$ represents the i-th cluster at the t-th experiment under clustering method ψ.

Theorem 1. *DAK is positive semi-definite.*

With Theorem 1 (see A.1 for the proof), we can define explicit feature maps in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ for the DAK as follows:

Definition 4 (Feature Map of DAK)**.** *Given the embeddings $\mathbf{X}_h^b$ of b-width h-hop slices, let $\mathbb{1}_\psi^{(t)} : \mathbf{X}_h^b \to \{0,1\}^{|\mathcal{C}_\psi^{(t)}|}$ be a cluster indicator of clustering method ψ at the t-th experiment. For any $\mathbf{x} \in \mathbf{X}_h^b$, $\mathbb{1}_\psi^{(t)}(\mathbf{x})_i = 1$ if and only if $\mathbf{x} \in \mathcal{C}_{\psi,i}^{(t)} \in \mathcal{C}_\psi^{(t)}$, otherwise $\mathbb{1}_\psi^{(t)}(\mathbf{x})_i = 0$. Then, the feature map ϕ_h^b of the DAK after T experiments is defined as:*

$$\phi_h^b(\mathbf{x}) = \frac{1}{\sqrt{T|\Psi|}} \, \|_{\psi \in \Psi} \|_{t=1}^{T} \, \mathbb{1}_\psi^{(t)}(\mathbf{x}) \tag{4}$$

where $\|$ means concatenation, $\phi_h^b(\mathbf{x}) \in \mathbb{R}^{T(\sum_{\psi \in \Psi} |\mathcal{C}_\psi|)}$ represents the concatenation of all $\mathbb{1}_\psi^{(t)}(\mathbf{x})$ for T experiments of ψ in the clustering method set Ψ.

Therefore, Equation 3 can be rewritten as:

$$\kappa_h^b(\mathbf{x}, \mathbf{y}) = \langle \phi_h^b(\mathbf{x}), \phi_h^b(\mathbf{y}) \rangle \tag{5}$$

Note that Equation 5 can directly compute the similarity between two slices extracted around two nodes residing in the same or different graphs. We extract the b-width h-hop slices around all the nodes in each graph. For the feature maps of DAK in each graph, we treat them as i.i.d. samples from an unknown distribution. Next, we utilize kernel mean embedding [Muandet *et al.*, 2017] to construct the Deep Graph Alignment Kernel (DGAK) between two graphs G_1 and G_2 from the perspective of distribution as follows:

$$\begin{aligned} \mathcal{K}_h^b(G_1, G_2) &= \langle \frac{1}{|\mathbf{X}_h^b|} \sum_{\mathbf{x} \in \mathbf{X}_h^b} \phi_h^b(\mathbf{x}), \frac{1}{|\mathbf{Y}_h^b|} \sum_{\mathbf{y} \in \mathbf{Y}_h^b} \phi_h^b(\mathbf{y}) \rangle \\ &= \frac{1}{|\mathbf{X}_h^b||\mathbf{Y}_h^b|} \sum_{\mathbf{x} \in \mathbf{X}_h^b} \sum_{\mathbf{y} \in \mathbf{Y}_h^b} \kappa_h^b(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{6}$$

where $\mathbf{X}_h^b$ and $\mathbf{Y}_h^b$ are the deep embeddings of b-width h-hop slices in graphs G_1 and G_2, respectively.

Finally, our Deep Hierarchical Graph Alignment Kernel (DHGAK) is defined as follows:

$$\mathcal{K}(G_1, G_2) = \sum_{h=1}^{H} \mathcal{K}_h^b(G_1, G_2) \tag{7}$$

Since the sum of positive semi-definite kernels is also positive semi-definite, both DGAK and DHGAK are positive semi-definite.

3.4 Theoretical Analysis of DHGAK

We explicit the transitivity of alignment operation in DAK, DGAK, and DHGAK at first. Given the set of clustering method Ψ, experiment times $T \in \mathbb{N}$ and $\mathbf{x}, \mathbf{y} \in \mathbf{X}_h^b$. Let the $\mathcal{M}_{h,\psi}^{b,t} : \mathbf{X}_h^b \times \mathbf{X}_h^b \to \{0, 1\}$ be the alignment operation at the t-th experiment under $\psi \in \Psi$, which satisfies:

$$\mathcal{M}_{h,\psi}^{b,t}(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ and } \mathbf{y} \text{ are assigned to the same} \\ & \text{cluster at the } t\text{-th experiment under } \psi; \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Therefore, DAK can be re-expressed as:

$$\kappa_h^b(\mathbf{x}, \mathbf{y}) = \frac{1}{T|\Psi|} \sum_{\psi \in \Psi} \sum_{t=1}^{T} \mathcal{M}_{h,\psi}^{b,t}(\mathbf{x}, \mathbf{y}) \tag{9}$$

DGAK and DHGAK can also be re-expressed in the sum of alignment operations. The theorem (See A.2 for the proof) below shows the alignment operation is transitive.

Theorem 2. *The alignment operation $\mathcal{M}_{h,\psi}^{b,t}$ in DAK, DGAK and DHGAK is transitive.*

Secondly, we care about the feasibility of using kernel mean embedding in Equation 6. We treat the feature maps of DAK in each graph as i.i.d. samples from an unknown distribution $\mathbb{G}$ and take kernel mean embedding of them to construct DGAK and DHGAK. When considering distribution classification tasks, Distributional Risk Minimization (DRM) [Muandet *et al.*, 2012] assures that any optimal solutions of any loss function can be expressed as a finite linear combination of the mean embedding of $\mathbb{G}_1, \ldots, \mathbb{G}_N$. DRM guarantees the feasibility of graph kernels derived by kernel mean embedding for graph classification problems.

Finally, Theorem 3 is introduced to assure that there exist some sets of clustering methods $\Psi_o \subset \mathfrak{M}$, derived from which the feature map of DHGAK is linearly separable in its RKHS for any dataset.

Theorem 3 (Linear Separation Property of DHGAK)**.** *Given a graph dataset $\mathcal{D} = \{(G_i, y_i)\}_{i=1}^N$, let Ψ_o denote the set of clustering method $\{\psi_j\}_{j=1}^{|\Psi_o|}$, where $\psi_j : \mathbf{X}_h^b \to \mathcal{C}_{\psi_j}$ separates each point $\mathbf{x}_h^b$ (the embedding of a slice extracted around a node v) to the cluster $\mathcal{C}_{\psi_j,k}$ if and only if $v \in \mathcal{V}_k$ belonging to G_k. For any $H, T \in \mathbb{N}$, the feature map of DHGAK is linearly separable for one class graphs $\{(G_p, y_p)\}$ from the other class graphs $\{(G_q, y_q) : y_q \neq y_p\}$.*

The proof of Theorem 3 is given in A.3. It's worth noticing that Ψ_o is only utilized for theoretical derivations. In addition, the theorem offers a rough estimator for the number of clusters, denoted as $|\mathcal{C}_\psi| \simeq |\mathcal{D}|$. This estimation is useful for the detailed implementations of DHGAK.

The worst-case time complexity of DHGAK is bounded by $\mathcal{O}(HN(n^2+ne)\times\mathcal{T}_{NLM}+HT|\Psi|\times\mathcal{T}_C(Nn))$, where $\mathcal{T}_{NLM}$ is the inference time complexity of the NLM per token, n, e are the average number of nodes and edges in graphs, and $\mathcal{T}_C(Nn)$ is the average time complexity of the clustering methods we select. Refer to Appendix B.3 for a more detailed analysis of the time complexity of DHGAK.

In short, DHGAK is proven to be alignment transitive, and theoretically, for any dataset, there exists a family of clustering methods derived from which DHGAK's feature maps are linearly separable in the RKHS.

4 Experimental Evaluation

In this section, we analyze the performance of DHGAK compared to some state-of-the-art graph kernels. First, we give the experimental setup and introduce two realizations of DHGAK. Next, we demonstrate the performance of DHGAK on graph classification, the parameter sensitivity analysis of DHGAK, and the ablation study to reveal the contribution of each component in DHGAK. Finally, we compare the running time of each graph kernel.

4.1 Experimental Setup

The generalization of DHGAK allows one to use any combination of Natural Language Models (NLM) for deep embedding and clustering methods for alignment to implement DHGAK. The NLMs we select are BERT [Devlin *et al.*, 2018] and word2vec [Mikolov *et al.*, 2013], and the clustering method we use is K-means. By doing so, we get two realizations of DHGAK: DHGAK-BERT and DHGAK-w2v. See C.2 and D.2 for more details of the NLMs and realizations.

We evaluate DHGAK on 16 real-world datasets downloaded from [Kersting *et al.*, 2016]. For datasets without node labels, we use the degree of each node as its node label. More details about datasets are provided in C.1. All experiments were conducted on a server equipped with a dual-core Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz, 256 GB memory, and Ubuntu 18.04.6 LTS with 6 RTX 3090 GPUs.

The parameters of DHGAK are set as follows. As discussed in Theorem 3, we set the number of clusters for different hops h as the same, around the size of the graph dataset, i.e., $|\mathcal{C}_\psi| = |\mathcal{D}| \times clusters\_factor$, where the $clusters\_factor$ are searched as base-10 log scale. We fix the experiment times T to 3 and other parameters are grid-searched as shown in Table 2 via 10-fold cross-validation on the training data.

We compare DHGAK with 9 state-of-the-art graph kernels, i.e., PM [Nikolentzos *et al.*, 2017], WL [Shervashidze *et al.*, 2011], WWL [Togninalli *et al.*, 2019], FWL [Schulz *et al.*, 2022], RetGK [Zhang *et al.*, 2018], WL-OA [Kriege *et al.*, 2016], GWL [Bause and Kriege, 2022], GAWL [Nikolentzos and Vazirgiannis, 2023], and GraphQNTK [Tang and Yan, 2022]. The parameters of these methods are set according to their original papers and implementations. We use the

Dataset	DHGAK-BERT	DHGAK-w2v	PM	WL	WWL	FWL	RetGK	WL-OA	GWL	GAWL	GraphQNTK
KKI	$\mathbf{58.9} \pm \mathbf{8.2}$	56.7 ± 5.2	52.3 ± 2.5	50.4 ± 2.8	55.4 ± 5.1	54.3 ± 19.8	48.5 ± 3.0	54.4 ± 12.8	$\underline{57.8 \pm 4.0^{\dagger}}$	56.8 ± 23.1	56.8 ± 22.9
MUTAG	$\mathbf{90.9} \pm \mathbf{4.2}$	$\underline{90.4 \pm 4.6}$	$86.7 \pm 0.6^{\dagger}$	$82.1 \pm 0.4^{\dagger}$	$87.3 \pm 1.5^{\dagger}$	85.7 ± 7.5	$90.3 \pm 1.1^{\dagger}$	$84.5 \pm 1.7^{\dagger}$	86.0 ± 1.2	$87.3 \pm 6.3^{\dagger}$	$88.4 \pm 6.5^{\dagger}$
PTC_MM	$\underline{70.5 \pm 4.6}$	$\mathbf{70.9} \pm \mathbf{5.6}$	63.4 ± 4.4	67.2 ± 1.6	69.1 ± 5.0	67.0 ± 5.1	$67.9 \pm 1.4^{\dagger}$	65.2 ± 6.4	65.1 ± 1.8	66.3 ± 5.3	66.4 ± 12.7
PTC_MR	$\mathbf{66.6} \pm \mathbf{7.7}$	63.7 ± 7.3	$60.2 \pm 0.9^{\dagger}$	61.3 ± 0.9	$\underline{66.3 \pm 1.2^{\dagger}}$	59.3 ± 7.3	$62.5 \pm 1.6^{\dagger}$	$63.6 \pm 1.5^{\dagger}$	59.9 ± 1.6	59.0 ± 4.4	59.4 ± 9.9
PTC_FM	$\underline{66.2 \pm 5.2}$	$\mathbf{66.5} \pm \mathbf{6.0}$	61.0 ± 3.9	64.4 ± 2.1	65.3 ± 6.2	61.0 ± 6.9	$63.9 \pm 1.3^{\dagger}$	62.5 ± 6.3	$62.6 \pm 1.9^{\dagger}$	63.9 ± 4.4	63.8 ± 5.3
PTC_FR	$\mathbf{72.1} \pm \mathbf{5.8}$	$\underline{71.3 \pm 4.9}$	67.8 ± 2.2	66.2 ± 1.0	67.3 ± 4.2	67.2 ± 4.6	$67.8 \pm 1.1^{\dagger}$	68.4 ± 4.3	66.0 ± 1.1	64.7 ± 3.6	68.9 ± 9.0
BZR	$\mathbf{89.4} \pm \mathbf{3.3}$	88.4 ± 4.5	81.5 ± 2.0	87.3 ± 0.8	80.5 ± 1.7	87.6 ± 5.0	$86.4 \pm 1.2^{\dagger}$	81.5 ± 4.7	87.7 ± 0.4	$\underline{89.1 \pm 4.1}$	83.1 ± 4.7
COX2	$\mathbf{84.6} \pm \mathbf{2.2}$	83.1 ± 4.0	78.2 ± 0.8	81.2 ± 1.1	83.1 ± 3.0	79.2 ± 2.9	$80.1 \pm 0.9^{\dagger}$	78.4 ± 1.1	82.9 ± 1.1	$\underline{83.5 \pm 3.1}$	79.5 ± 1.6
DHFR	$\mathbf{85.5} \pm \mathbf{4.4}$	$\underline{84.0 \pm 4.6}$	76.9 ± 3.9	82.4 ± 0.9	82.7 ± 3.5	82.8 ± 5.8	$81.5 \pm 0.9^{\dagger}$	82.7 ± 4.5	80.6 ± 0.8	83.7 ± 3.0	72.1 ± 4.8
ENZYMES	$\mathbf{65.3} \pm \mathbf{7.9}$	57.3 ± 5.4	$40.3 \pm 0.3^{\dagger}$	$52.2 \pm 1.3^{\dagger}$	$59.1 \pm 0.8^{\dagger}$	51.8 ± 5.5	$\underline{60.4 \pm 0.8^{\dagger}}$	$59.9 \pm 1.1^{\dagger}$	54.5 ± 1.6	58.5 ± 4.8	35.8 ± 5.3
PROTEINS	$\mathbf{76.6} \pm \mathbf{4.3}$	75.2 ± 4.6	74.8 ± 3.9	75.5 ± 0.3	$74.3 \pm 0.6^{\dagger}$	74.6 ± 3.8	$75.8 \pm 0.6^{\dagger}$	$\underline{76.4 \pm 0.4^{\dagger}}$	73.7 ± 0.5	$74.7 \pm 3.0^{\dagger}$	$71.1 \pm 3.2^{\dagger}$
DD	$\underline{81.0 \pm 4.7}$	80.1 ± 3.1	$77.8 \pm 0.5^{\dagger}$	$79.8 \pm 0.4^{\dagger}$	$79.7 \pm 0.5^{\dagger}$	78.4 ± 2.4	$\mathbf{81.6} \pm \mathbf{0.3}^{\dagger}$	$79.2 \pm 0.4^{\dagger}$	$79.0 \pm 0.8^{\dagger}$	$78.7 \pm 2.8^{\dagger}$	79.6 ± 4.7
NCI1	$\underline{85.9 \pm 2.0}$	85.2 ± 2.1	$72.9 \pm 0.5^{\dagger}$	$82.2 \pm 0.2^{\dagger}$	$85.8 \pm 0.3^{\dagger}$	85.4 ± 1.6	$84.5 \pm 0.2^{\dagger}$	$\mathbf{86.1} \pm \mathbf{0.2}^{\dagger}$	$85.3 \pm 0.4^{\dagger}$	$\underline{85.9 \pm 1.2^{\dagger}}$	$77.2 \pm 2.7^{\dagger}$
IMDB-B	$\mathbf{75.3} \pm \mathbf{2.7}$	73.5 ± 1.9	74.8 ± 2.3	73.8 ± 3.9	73.3 ± 4.1	72.0 ± 4.6	$71.9 \pm 1.0^{\dagger}$	$\underline{75.2 \pm 2.8}$	$73.7 \pm 1.3^{\dagger}$	$74.5 \pm 4.1^{\dagger}$	$73.3 \pm 3.6^{\dagger}$
IMDB-M	$\mathbf{52.1} \pm \mathbf{2.4}$	50.0 ± 2.3	50.7 ± 2.8	51.1 ± 2.7	50.4 ± 3.4	50.0 ± 2.9	$47.7 \pm 0.3^{\dagger}$	50.9 ± 2.8	50.6 ± 0.5	$\underline{51.7 \pm 5.2^{\dagger}}$	$48.1 \pm 4.3^{\dagger}$
COLLAB	$\mathbf{81.7} \pm \mathbf{1.5}$	$\underline{81.6 \pm 1.7}$	80.5 ± 1.5	78.7 ± 1.5	79.1 ± 1.5	78.1 ± 1.2	$81.0 \pm 0.3^{\dagger}$	$80.7 \pm 0.1^{\dagger}$	80.6 ± 0.3	$\underline{81.5 \pm 2.0^{\dagger}}$	$> 24h$

Table 1: 10-fold cross-validation C-SVM classification accuracy (%). † means the result is adopted from the original papers (the highest accuracy marked in **bold** and the second highest accuracy marked as underlined).

implementation of PM, WL, and WL-OA from GraKeL library [Siglidis *et al.*, 2020]. All kernel matrices are normalized as $K(i,j) = K(i,j)/\sqrt{K(i,i)K(j,j)}$. We use 10-fold cross-validation with a binary C-SVM [Chang and Lin, 2011] to test the classification performance of each graph kernel. The parameter C for each fold is independently tuned from $\{10^{-3}, 10^{-2}, \ldots, 10^{4}\}$ using the training data from that fold. We report the average classification accuracy and standard deviation over the 10-fold.

parameter	search ranges
(BERT) fine-tune epochs	$\{0, 3\}$
b	$\{0, 1, 2\}$
H	$\{1, 3, \ldots, 9\}$
α	$\{0, 0.2, \ldots, 1\}$
$clusters\_factor$	$\{0.1, 0.1 \times r, \ldots, 0.1 \times r^{8}, 2\}$

Table 2: Parameter grid ranges for datasets. $r = 10^{(\frac{\log_{10} 2}{0.1})}$ represents the interval of base-10 log scale ranges. (BERT) represents the parameter only available for DHGAK-BERT.

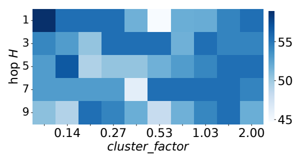
4.2 Classification Results

Classification Accuracy

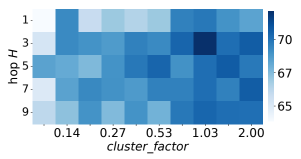
The classification accuracy results of two realizations of DHGAK (DHGAK-BERT and DHGAK-w2v) are shown in Table 1. On 16 datasets, both DHGAK-BERT and DHGAK-w2v show superior performance compared to all the competitors on 14 datasets except DD and NCI1. For small datasets (datasets above ENZYMES in Table 1), DHGAK-BERT and DHGAK-w2v outperform all the baselines on all the datasets. For large datasets (datasets below PROTEINS in Table 1), DHGAK-w2v shows a slight drop in classification accuracy and is comparable to GAWL; while DHGAK-BERT outperforms all baselines on these datasets except DD and NCI1, where it secures the second-best performance. More specifically, DHGAK significantly outperforms some baselines on some datasets. For example, on DHFR, PTC_MM, PTC_FR, and ENZYMES, DHGAK exhibits absolute improvements of 1.8%, 1.8%, 3.2%, and 4.9%, respectively, over other methods. Overall, DHGAK achieves the best performance on 14 of 16 datasets and the second best performance on DD and NCI1.

Parameter Sensitivity

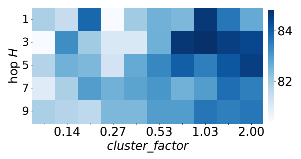
In this section, we study the parameter sensitivity of DHGAK. The parameters are: the number of clusters $|\mathcal{C}_{\psi}| = |\mathcal{D}| \times cluster\_factor$, the maximum hop H, the slice width b, decay weight $\alpha \in [0, 1]$ and experiment times T (See D.1). We run DHGAK-BERT with 3 fine-tuned epochs on four datasets DHFR, PTC_FR, KKI, and IMDB-B from different categories.



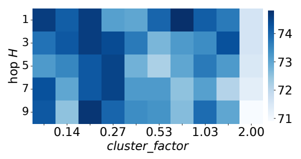
(a) KKI



(b) PTC_FR



(c) DHFR

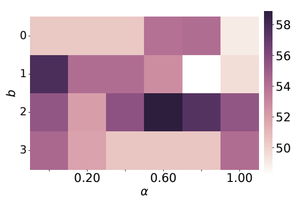


(d) IMDB-B

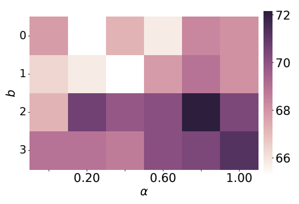
Figure 4: Paramter sensitivity analysis on H and $cluster\_factor$ in DHGAK-BERT. We present the classification accuracy for combinations of $H \in \{1, 3, \ldots, 9\}$ and $cluster\_factor \in \{0.1, 0.14, \ldots, 2\}$ selected from 0.1 to 2 in base 10 log scale.

With b fixed to the best parameter in each dataset, and α fixed to 0.6, we draw the heatmap of average classification accuracy over the 10 folds for all combinations of $H \in \{1, 3, \ldots, 9\}$ and $cluster\_factor$ varying from 0.1 to 2 in base 10 log scale. In Figure 4, We can observe that DHGAK-BERT behaves stably across various parameter combinations, consistently achieving high performance.

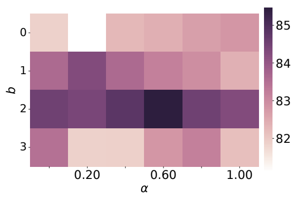
Then we fix H and $|\mathcal{C}_{\psi}|$ to the best parameter, and vary $b \in \{0, 1, 2, 3\}$ and $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Figure 5 shows the heatmap for all combinations of b and α. As can be



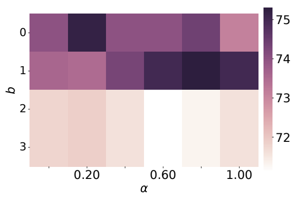
(a) KKI



(b) PTC_FR



(c) DHFR



(d) IMDB-B

Figure 5: Parameter sensitivity analysis on b and α in DHGAK-BERT. We present the classification accuracy for combinations of $b \in \{0, 1, 2, 3\}$ and $\alpha \in \{0, 0.2, 0.4, \ldots, 1.0\}$.

seen, the best classification accuracy may be achieved when using values from the set $\{0, 1, 2\}$ for b, with α approximately around 0.6. For the IMDB-B dataset, when α is fixed, increasing b beyond 2 does not improve the performance. The reason is that, for datasets with a large number of edges in the graph, the encoding of slice can easily exceed the input sentence length limit (the token size) of BERT, leading to a truncation of encoding. Exploiting Large Language Models (LLM) such as GPT4 is left for future work.

4.3 Ablation Study

In this section, we compare DHGAK-BERT to its three variants: HGAK-label, DGAK-BERT, and DHW-BERT. HGAK-label is a graph kernel whose label embedding function g defined in Definition 2 is one-hot node label mapping. DGAK-BERT is a graph kernel whose deep embeddings are learned from the nodes' $(H + b)$-depth truncated BFS trees, not from hierarchical slices. DHW-BERT uses the 1-Wasserstein distance instead of using DAK and DGAK, whose kernel matrix is computed by summing the Laplacian kernel of 1-Wasserstein distances between graphs for each hop. The average classification accuracy via 10-fold cross-validation of them is given in Table 3.

The effectiveness of each component in DHGAK is demonstrated in the table. Comparisons between the performance of the three ablation variants show that HGAK-label ranks the first, indicating that the deep embedding captures the neighborhood information of slice better than direct one-hot encoding. DHW-BERT ranks the second, indicating that DAK has a better ability to capture the inherent connections of slices compared to the Wasserstein distance. Finally, DGAK-BERT exhibits the lowest performance, suggesting that the slice structure we introduced significantly contributes to the model performance of DHGAK compared to the common BFS trees.

Dataset	DHGAK-BERT	HGAK-label	DGAK-BERT	DHW-BERT
KKI	$\mathbf{58.9} \pm \mathbf{8.2}$	55.4 ± 5.2	55.4 ± 5.2	49.3 ± 12.2
MUTAG	$\mathbf{90.9} \pm \mathbf{4.2}$	88.2 ± 4.6	82.9 ± 6.7	86.1 ± 4.4
PTC_MM	$\mathbf{70.5} \pm \mathbf{4.6}$	66.4 ± 6.4	67.2 ± 4.9	66.2 ± 5.7
PTC_FR	$\mathbf{72.1} \pm \mathbf{5.8}$	70.3 ± 5.3	66.1 ± 5.9	67.8 ± 3.6
DHFR	$\mathbf{85.5} \pm \mathbf{4.4}$	82.2 ± 4.9	82.6 ± 4.8	84.3 ± 5.6
ENZYMES	$\mathbf{65.3} \pm \mathbf{7.9}$	57.3 ± 8.2	58.8 ± 9.0	60.3 ± 5.2
IMDB-B	$\mathbf{75.3} \pm \mathbf{2.7}$	74.7 ± 4.0	70.6 ± 1.2	73.3 ± 2.2
IMDB-M	$\mathbf{52.1} \pm \mathbf{2.4}$	51.7 ± 2.3	50.7 ± 3.9	51.6 ± 2.8
COLLAB	$\mathbf{81.7} \pm \mathbf{1.5}$	81.5 ± 1.0	81.2 ± 1.8	$> 24h$
Avg. rank	1.0	2.6	3.3	3.0

Table 3: Comparison of the average classification accuracy of DHGAK-BERT, HGAK-label, DGAK-BERT, DHW-BERT. Note that DHW-BERT applying Wasserstein distance instead of DAK fails to complete in 24 hours on large datasets.

Method	PTC_MM	PROTEINS	DD	COLLAB
DHGAK-BERT	11.58	754.09	5456.99	2452.41
DHGAK-w2v	0.91	130.81	2708.16	1986.64
PM	2.38	23.62	1068.34	1262.27
WL	0.74	1.52	14.81	82.10
WWL	13.63	238.67	4953.69	10044.34
FWL	7.34	960.94	2531.51	7405.03
WL-OA	1.70	174.83	6090.02	30743.83
GAWL	2.02	31.12	519.69	3854.78
GraphQNTK	27.47	773.25	6434.50	>24h

Table 4: Running time (in seconds) comparison between two realizations of DHGAK and some baselines. DHGAK-w2v is comparable to PM. DHGAK-BERT is faster than WWL, WL-OA, FWL, and GraphQNTK on large datasets.

4.4 Running Time

In Table 4, we demonstrate the running time (in seconds) of two realizations of DHGAK compared with some baselines on four datasets. The datasets are sorted by their sizes. As shown in the table, the fastest GKs are $\mathcal{R}$-convolution graph kernels WL and GAWL. The running time of DHGAK-w2v is comparable to PM. DHGAK-BERT is much slower than DHGAK-w2v, but it is still faster than WWL, WL-OA, FWL, and GraphQNTK on large datasets such as COLLAB. We also find that the efficiency of DHGAK behaves better on large datasets. It is evident that time bottleneck of DHGAK-BERT lies in the inference process of BERT.

5 Conclusion

In this paper, we introduce Deep Hierarchical Graph Alignment Kernels (DHGAK), a novel framework to address the prevalent issue of neglecting relations among substructures in existing $\mathcal{R}$-convolution graph kernels. We propose a positive semi-definite kernel called Deep Alignment Kernel (DAK), which efficiently aligns relational graph substructures using clustering methods on the deep embeddings of substructures learned by Natural Language Models. Then, the Deep Graph Alignment Kernel (DGAK) is constructed by applying kernel mean embedding on the feature maps of DAK. Finally, DHGAK is constructed by the summation of DGAK on slices of different hierarchies. The theoretical analysis demonstrates the effectiveness of our approach, and experimental results indicate that the two realizations of DHGAK outperform the state-of-the-art graph kernels on most datasets.

Acknowledgments

We thank the anonymous reviewers for their valuable and constructive comments. This work was supported partially by the National Natural Science Foundation of China (grants # 62176184 and 62206108), the National Key Research and Development Program of China (grant # 2020AAA0108100), and the Fundamental Research Funds for the Central Universities of China.

References

[Bai *et al.*, 2022] Lu Bai, Lixin Cui, and Hancock Edwin. A hierarchical transitive-aligned graph kernel for unattributed graphs. In *International Conference on Machine Learning*, pages 1327–1336. PMLR, 2022.

[Bause and Kriege, 2022] Franka Bause and Nils Morten Kriege. Gradual weisfeiler-leman: Slow and steady wins the race. In *Learning on Graphs Conference*, pages 20–1. PMLR, 2022.

[Bonacich, 1987] Phillip Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

[Borgwardt and Kriegel, 2005] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[Borgwardt *et al.*, 2005] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Naumovich Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[Debnath *et al.*, 1991] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dobson and Doig, 2003] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[Du *et al.*, 2019] Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

[Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[Gretton *et al.*, 2012] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[Haussler and others, 1999] David Haussler et al. Convolution kernels on discrete structures. Technical report, Citeseer, 1999.

[Helma *et al.*, 2001] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

[Kersting *et al.*, 2016] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016.

[Kriege *et al.*, 2016] Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. *Advances in neural information processing systems*, 29, 2016.

[Lazebnik *et al.*, 2006] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[Leman and Weisfeiler, 1968] AA Leman and Boris Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

[MacQueen and others, 1967] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Muandet *et al.*, 2012] Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. *Advances in neural information processing systems*, 25, 2012.

[Muandet *et al.*, 2017] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[Nikolentzos and Vazirgiannis, 2023] Giannis Nikolentzos and Michalis Vazirgiannis. Graph alignment kernels using weisfeiler and leman hierarchies. In *International Conference on Artificial Intelligence and Statistics*, pages 2019–2034. PMLR, 2023.

[Nikolentzos *et al.*, 2017] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[Pan *et al.*, 2016] Shirui Pan, Jia Wu, Xingquan Zhu, Guodong Long, and Chengqi Zhang. Task sensitive feature exploration and learning for multitask graph classification. *IEEE transactions on cybernetics*, 47(3):744–758, 2016.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[Schulz *et al.*, 2022] Till Schulz, Pascal Welke, and Stefan Wrobel. Graph filtration kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8196–8203, 2022.

[Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[Siglidis *et al.*, 2020] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *The Journal of Machine Learning Research*, 21(1):1993–1997, 2020.

[Sutherland *et al.*, 2003] J. J. Sutherland, L. A. O'Brien, and D. F. Weaver. Spline-fitting with a genetic algorithm: a method for developing classification structure-activity relationships. *J Chem Inf Comput Sci*, 43(6):1906–15, 2003.

[Tang and Yan, 2022] Yehui Tang and Junchi Yan. Graphqntk: Quantum neural tangent kernel for graph data. *Advances in Neural Information Processing Systems*, 35:6104–6118, 2022.

[Togninalli *et al.*, 2019] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Advances in neural information processing systems*, 32, 2019.

[Vishwanathan *et al.*, 2010] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[Wale *et al.*, 2008] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14:347–375, 2008.

[Yanardag and Vishwanathan, 2015a] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[Yanardag and Vishwanathan, 2015b] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[Ye *et al.*, 2019] Wei Ye, Zhen Wang, Rachel Redberg, and Ambuj Singh. Tree++: Truncated tree based graph kernels. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1778–1789, 2019.

[Ye *et al.*, 2023] Wei Ye, Hao Tian, and Qijun Chen. Multiscale wasserstein shortest-path graph kernels for graph classification. *IEEE Transactions on Artificial Intelligence*, 2023.

[Zhang *et al.*, 2018] Zhen Zhang, Mianzhi Wang, Yijian Xiang, Yan Huang, and Arye Nehorai. Retgk: Graph kernels based on return probabilities of random walks. *Advances in Neural Information Processing Systems*, 31, 2018.