AKBR: Learning Adaptive Kernel-based Representations for Graph Classification

Lu Bai[1] , **Feifei Qian**[1] , **Lixin Cui**[2*] , **Ming Li**[3,4] , **Hangyuan Du**[5] , **Yue Wang**[2] , **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 Institute of Optoelectronics, Jinhua, China;
[4]Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, China;
[5]School of Computer and Information Technology, Shanxi University, Taiyuan, China;
[6]Department of Computer Science, University of York, York, United Kingdom.
bailu@bnu.edu.cn, feifei_qian@mail.bnu.edu.cn, cuilixin@cufe.edu.cn

Abstract

In this paper, we propose a new model to learn Adaptive Kernel-based Representations (AKBR) for graph classification. Unlike state-of-the-art R-convolution graph kernels that are defined by merely counting any pair of isomorphic substructures between graphs and cannot provide an end-to-end learning mechanism for the classifier, the proposed AKBR approach aims to define an end-to-end representation learning model to construct an adaptive kernel matrix for graphs. To this end, we commence by leveraging a novel feature-channel attention mechanism to capture the interdependencies between different substructure invariants of original graphs. The proposed AKBR model can thus effectively identify the structural importance of different substructures, and compute the R-convolution kernel between pairwise graphs associated with the more significant substructures specified by their structural attentions. Furthermore, the proposed AKBR model employs all sample graphs as the prototype graphs, naturally providing an end-to-end learning architecture between the kernel computation as well as the classifier. Experimental results show that the proposed AKBR model outperforms existing state-of-the-art graph kernels and deep learning methods on standard graph benchmarks.

1 Introduction

Graph-based representations are powerful tools for encapsulating structured data characterized by pairwise relationships among its components [Zambon *et al.*, 2018], and have been widely employed in various research fields, such as the analysis of social networks [Bai *et al.*, 2020b], financial transactions [Bai *et al.*, 2020a] and biological networks [Gasteiger *et al.*, 2021]. The main challenge arising in graph data analysis is how to learn representative numeric characteristics for discrete graph structures. One of the most effective methods for learning graph-structured data is to employ graph kernels.

Broadly speaking, graph kernels aim to describe the structural information in a high-dimension Hilbert space, typically defining a positive definite similarity measure between graphs. [Haussler and et al., 1999] proposed a generic way, namely the R-convolution framework, to define graph kernels. This is achieved by decomposing two graphs into substructures and evaluating the similarity between the pairs of substructures. Specifically, given two sample graphs G_p and G_q, assume $\mathbf{S} = \{g_1, ..., g_M\}$ is the set of their all possible substructures based on a specified graph decomposing approach, the R-convolution kernel K_R between G_p and G_q is defined as

$$K_R(G_p, G_q) = \sum_{\vec{g_p} \in \mathbf{S}} \sum_{\vec{g_q} \in \mathbf{S}} k(\vec{g_p}, \vec{g_q}), \tag{1}$$

where the function k is defined as the Dirac kernel, and $k(\vec{g_p}, \vec{g_q})$ is equal to 1 if the substructures $\vec{g_q}$ and $\vec{g_q}$ are isomorphic to each other, and 0 otherwise.

In recent years, the R-convolution framework has proven to be an effective way to define novel graph kernels and most state-of-the-art R-convolution graph kernels can be categorized into three main categories, i.e., the R-convolution kernels based on the walks, paths, and subgraph or subtree structures. For instance, the Random Walk Graph Kernel (RWGK) is proposed by [Gärtner *et al.*, 2003] based on the similarity measures between random walks. However, the random walks suffer from the notorious tottering problem and allow the repetitive visiting of vertices, leading to significant information redundancy for the RWGK kernel. [Borgwardt and Kriegel, 2005] have proposed a Shortest Path Graph Kernel (SPGK) by counting the pairs of shortest paths with the same length. Since the shortest paths are typically non-backtrack paths and can be computed in a polynomial time, the SPGK kernel can significantly overcome the drawbacks of the RWGK kernel. To capture more structural information, thus some subgraph-based or subtree-based R-convolution graph kernels have been developed. For instance, [Shervashidze *et al.*, 2009] have proposed a Graphlet Count Graph

*Corresponding Author: Lixin Cui

Kernel (GCGK) by counting the frequency of graphlet subgraphs of sizes 3, 4 and 5. Since the GCGK kernel cannot accommodate the vertex attributes, [Shervashidze *et al.*, 2011] have further developed the Weisfeiler-Lehman Subtree Kernel (WLSK) based on subtree invariants. Specifically, the WLSK kernel is defined by counting the number of pairwise isomorphic subtrees through an iterative aggregation of node neighbor labels, effectively enabling labeled graph classification. Moreover, since the WLSK kernel can efficiently and gradually aggregate the local topological substructure information (i.e., the vertex labels corresponding to subtree invariants) between neighbor vertices to further extract subtrees of large sizes, this kernel not only has better computational efficiency but also has superior effectiveness for graph classification, being one of the most popular graph kernels by now. Other graph kernels based on the R-convolution also include (1) the Wasserstein Weisfeiler-Lehman Subtree Kernel [Togninalli *et al.*, 2019], (2) the Subgraph Alignment Kernel [Kriege and Mutzel, 2012], (3) the Message Passing Graph Kernels [Nikolentzos and Vazirgiannis, 2018] etc.

Although state-of-the-art R-convolution graph kernels have demonstrated their performance on graph classification tasks, they still suffer from three common problems. First, these R-convolution graph kernels only focus on measuring the similarity or the isomorphism between all pairs of substructures, completely disregarding the importance of different substructures. As a result, some redundant structural information that is unsuitable for graph classification may also be considered. Second, these R-convolution graph kernels focus solely on the similarity between each pair of graphs, neglecting the common patterns shared among all sample graphs. Third, all these R-convolution graph kernels tend to employ the C-SVM classifier [Cortes and Vapnik, 1995] for classification, and the phase of training the classifier is entirely separated from that of the kernel construction. This limitation may restrict the performance of existing R-convolution graph kernels. To overcome the first shortcoming, [Aziz *et al.*, 2020] have employed the feature selection method to discard redundant substructure patterns associated with zero for the GCGK kernel, significantly improving the classification performance. However, this kernel method requires manually enumerating all possible graphlet substructure sets to compute the mean and variance, and still cannot provide an end-to-end learning framework to adaptively compute the kernel-based similarity. Overall, defining effective kernel-based approaches for graph classification remains challenging.

The objective of this work is to address the drawbacks of the aforementioned R-convolution graph kernels, by developing a novel framework to compute the Adaptive Kernel-based Representations (AKBR) for graph classification. One key innovation of the proposed AKBR model is that it can provide an end-to-end kernel-based learning framework to discriminate significant substructures and thus compute an adaptive kernel matrix between graphs. The main contributions are summarized as threefold.

First, to resolve the problem of ignoring the importance of different substructures that arise in existing R-convolution graph kernels, we propose to employ the attention mechanism as a means of feature selection to assign different weights to the substructures represented as features. We model the interdependency of different substructure-based features using the feature-channel attention mechanism to focus on the essential part of the substructure-based feature vectors of graphs.

Second, inspired by the graph dissimilarity or similarity embedding method presented by [Bunke and Riesen, 2008; Bai *et al.*, 2013], the resulting kernel matrix can be seen as a kind of kernel-based similarity embedding vectors of all sample graphs. As a result, the proposed AKBR model can adaptively discriminate the structural importance of different substructures, and further compute the adaptive kernel-based representations for graph classification, significantly overcoming the three aforementioned theoretical drawbacks arising in existing R-convolution graph kernels.

Third, we evaluate the proposed AKBR model on graph classification tasks. The experimental results demonstrate that the proposed model can significantly outperform state-of-the-art graph kernels and graph deep learning methods.

2 Related Works

In this section, we review some state-of-the-art R-convolution kernels and some classical Graph Neural Networks (GNNs). Besides, we theoretically analyze the drawbacks arising in these existing approaches, enlightening the proposed method.

2.1 Classical R-convolution Kernels

We briefly review two classical R-convolution kernels, including WLSK [Shervashidze *et al.*, 2011] and the SPGK [Borgwardt and Kriegel, 2005]. We commence by introducing the definition of the WLSK kernel that focuses on aggregating the structural information from neighboring vertices iteratively to capture1 subtree invariants through the classical Weisfeiler-Lehman Subtree-Invariant (WL-SI) method [Weisfeiler and Lehman, 1968]. Given two sample graphs G_p and G_q, assume $l_0(u)$ represents the initial label of vertex u. Specifically, for unlabeled graphs, the degree of each vertex is considered as the initial label. Then, for each iteration i, the WLSK constructs the multi-set label $\mathcal{L}^i_{\mathcal{N}}$ for each vertex u by aggregating and sorting the labels of u as well as its neighborhood vertices, i.e.,

$$\mathcal{L}^i_{\mathcal{N}}(u) = \mathrm{sort}(\{l_{i-1}(v)|v \in \mathcal{N}(u)\}), \tag{2}$$

where $\mathcal{N}(u)$ is the set of the neighborhood vertices of u. The WLSK kernel merges the multi-set label $\mathcal{L}^i_{\mathcal{N}}$ of each vertex u and into a new label $l_i(u)$ through a Hash function as

$$l_i(u) = \mathrm{Hash}(l_{i-1}(u), \mathcal{L}^{i-1}_{\mathcal{N}}(u)), \tag{3}$$

where Hash is the hash mapping function that relabels $\mathcal{L}^i_{\mathcal{N}}$ as a new single positive integer, and each $l_i(u)$ corresponds to a subtree rooted at u of height i. The iteration i ends when the number of iterations is met to the largest one (i.e., $I_{\max}$). The WLSK kernel $K_{\mathrm{WL}}(G_p, G_q)$ between the pair of graphs G_p and G_q can be defined by counting the number of shared pairwise isomorphic subtrees corresponding by $l_i(u)$, i.e.,

$$K_{\mathrm{WL}}(G_p, G_q) = \sum_{i=0}^{I_{\max}} \sum_{j=0}^{|\mathcal{L}^i|} \mathcal{N}(G_p, l^j_i)\mathcal{N}(G_q, l^j_i), \tag{4}$$

where $I_{\max}$ denotes the maximum number of the iteration i, $l_i^j \in \mathcal{L}^i$ is the j-th vertex label of $\mathcal{L}^i$, and $\mathcal{N}(G_p, l_i^j)$ represents the number of the subtrees corresponded by the label l_i^j and appearing in G_p.

The idea of the SPGK kernel is to compare the similarity between a pair of graphs by counting the number of shared shortest paths with the same lengths. The first step of computing the SPGK kernel is to extract all shortest paths from each graph by using the classical Floyd algorithm [Floyd, 1962]. Given the pair of graphs G_p and G_q, the SPGK is defined as

$$K_{\mathrm{SP}}(G_p, G_q) = \sum_{s_i \in \mathcal{S}} \mathcal{N}(G_p, s_i)\mathcal{N}(G_q, s_i), \qquad (5)$$

where $s_i \in \mathcal{S}$ is the shortest path of length i, $\mathcal{S}$ is the set of all possible shortest paths appearing in all graphs, and $\mathcal{N}(G_p, s_i)$ represents the number of s_i appearing in G_p.

Remarks. Although the WLSK and SPGK kernels associated with the C-Support Vector Machine (C-SVM) [Chang and Lin, 2011] have effective performance for graph classification, they still suffer from the theoretical drawbacks we discussed in Section 1.

2.2 Classical Graph Neural Networks

One way to overcome the aforementioned problems of R-convolution graph kernels is to adopt the GNN models, that are developed by generalizing the classical Neural Networks to the graph domain based on the spectral or spatial strategy [Bai *et al.*, 2022b; Cui *et al.*, 2024]. Since the spectral-based GNN models usually require that the graphs should have the same size, and are only suitable for node classification [Bruna *et al.*, 2014]. The spatial-based GNN models are widely developed for graph classification. Under this scenario, the Graph Isomorphism Network (GIN) proposed by [Xu *et al.*, 2019] exhibits the same expressive power with the WLSK kernel. The Diffusion Convolution Neural Network (DCNN) [Atwood and Towsley, 2016] uses different weights to propagate the neighborhood information from different hops to the center. The Graph Convolution Neural Network (PATCHY-SAN) [Niepert *et al.*, 2016] extracts the features by converting the graph structure into fixed-sized patches. The Deep Graph Convolution Neural Network (DGCNN) [Zhang *et al.*, 2018] sorts the nodes based on the substructure information extracted from the last graph convolutional layer and preserves predetermined numbers of nodes, resulting in the fixed-sized grid structure for the traditional convolution operation.

Remarks. Although the above GNNs can naturally provide an end-to-end framework, it is not available for classical R-convolution kernels. These GNNs still suffer from some similar drawbacks. First, the GIN and DCNN models tend to directly sum up the local vertex features as the global graph representation through the SumPool operation, discarding the importance of the different local structure information residing on different nodes. Second, the PATCHY-SAN and DCNN models only preserve the local structure information residing on the top-ranked nodes based on the SortPool operation, resulting in significant information loss. In fact, these drawbacks also appear in other alternative GNNs, influencing the performance of graph classification.

Several works have also explored the intersection between graph kernels and graph-based deep learning methods. Specifically, GNTK [Du *et al.*, 2019] bridges graph kernels and graph neural networks (GNNs) by modeling smooth functions defined over graphs. KerGNN [Feng *et al.*, 2022] introduces trainable hidden graphs as filters to refine node embeddings, while RWNN [Nikolentzos and Vazirgiannis, 2020] utilizes random walk kernels to measure similarities between input graphs and hidden structures. However, these approaches generally require increased parameters, thereby incurring higher computational costs.

3 The Proposed AKBR Model for Graphs

In this section, we develop a novel Adaptive Kernel-based Representations (AKBR) model. We introduce the detailed definition of the proposed AKBR model.

3.1 The Framework of the AKBR Model

We commence by defining the framework of the proposed AKBR model. Specifically, the computational architecture is shown in Figure 1, mainly consisting of three procedures. **First**, we construct the feature vector $\varphi(G_i)$ for each sample graph G_i based on the substructure invariants extracted with a specific R-convolution graph kernel. In this work, we propose to adopt the classical WLSK and SPGK kernels for the proposed framework. This is because the subtree and shortest path substructures are effective in representing the structural characteristics of the original graphs. **Second**, we employ an attention layer to assign different weights to different substructures, and the critical features will be associated with larger weights through the attention mechanism. **Third**, the resulting kernel matrix between pairwise graphs can be computed as the dot product between their attention-based substructure feature vectors. Inspired by the graph dissimilarity or similarity embedding method presented by [Bunke and Riesen, 2008; Bai *et al.*, 2013], we employ the resulting kernel matrix as the kernel-based similarity embedding vectors of all sample graphs.

As a result, the framework of the proposed AKBR model can provide an end-to-end learning architecture between the kernel computation as well as the classifier, i.e., the proposed AKBR model can adaptively compute the kernel matrix associated with the most effective substructure invariants.

3.2 The Definition of the AKBR Model

In this subsection, we provide a detailed explanation of the computation process for each step.

The Construction of Substructure Invariants

We employ the classical WLSK and SPGK kernels to extract the subtrees and the shortest paths as the substructure invariants. For the WLSK kernel, the subtree-based feature vector $\varphi_{\mathrm{WL}}(G)$ of a sample graph G is defined as

$$\varphi_{\mathrm{WL}}(G) = [\mathrm{n}(G, l_1), \ldots, \mathrm{n}(G, l_i), \ldots, \mathrm{n}(G, l_{|\mathcal{L}|})], \qquad (6)$$

where l_i is the vertex label defined by Eq.(3) and corresponds to a subtree invariant, each element $\mathrm{n}(G, l_i)$ is the number of

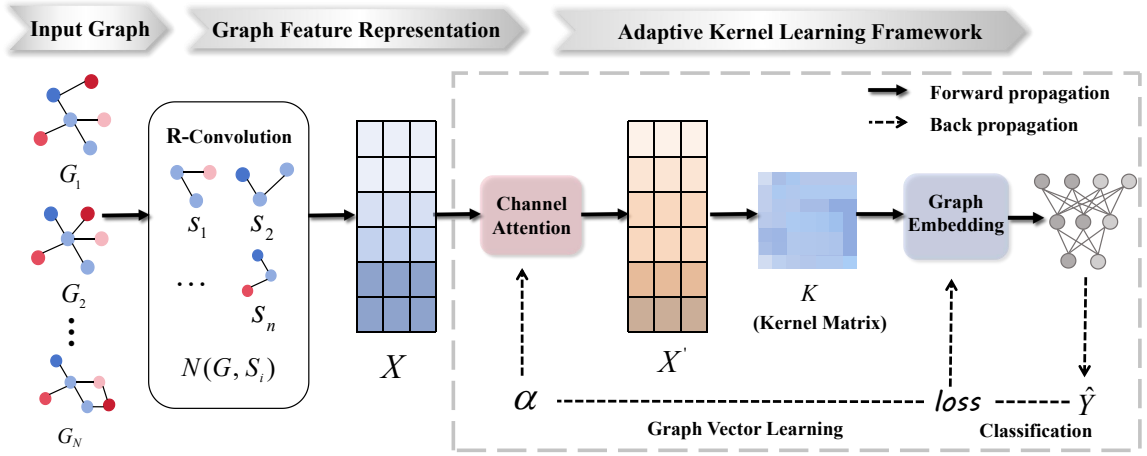


Figure 1: The Framework of the Proposed AKBR Model.

the corresponding subtree invariants appearing in G, and $|\mathcal{L}|$ is a positive integer and refers to the number of all distinct subtree invariant labels. Similarly, for the SPGK kernel, the feature vector $\varphi_{\mathrm{SP}}(G)$ of the graph G is defined as

$$\varphi_{\mathrm{SP}}(G) = [\mathrm{n}(G, s_1), \ldots, \mathrm{n}(G, s_i), \ldots, \mathrm{n}(G, s_{|S|})], \quad (7)$$

where each element $\mathrm{n}(G, s_i)$ is the number of the shortest paths with the length s_i in the graph G, and $|S|$ denotes the greatest length of the shortest paths over all graphs. With the substructure-based feature vectors of all graphs in $\mathbf{G} = \{G_1, \ldots, G_N\}$ to hand, we can derive the feature matrix $\mathbf{X} \in \mathbb{R}^{N \times L}$ for the entire graph dataset $\mathbf{G}$, i.e.,

$$\mathbf{X}_{(\cdot)} = \begin{pmatrix} \varphi_{(\cdot)}(G_1) \\ \cdots \\ \varphi_{(\cdot)}(G_j) \\ \cdots \\ \varphi_{(\cdot)}(G_N) \end{pmatrix}, \quad (8)$$

where N denotes the number of graphs, L denotes the dimension of each feature vector $\varphi(G_j)$ for the graph $G_j \in \mathbf{G}$, and $(\cdot)$ corresponds to either the WLSK or the SPGK kernel.

Attention Mechanism for Feature Selection

As we have stated previously, some substructure-based features may be more prevalent in some graphs. These features naturally encapsulate more significant and discriminative structural information for classification. Thus, assigning a more substantial weight to these features is preferred. As a result, it is necessary to perform a thorough examination to evaluate the importance of different features across all graphs. To this end, we propose to employ the attention mechanism as a means of feature selection, and adaptively identify the most effective features of the feature matrix $\mathbf{X}_{(\cdot)} \in \mathbb{R}^{N \times L}$.

There have been various types of attention mechanisms, including the self-attention [Vaswani *et al.*, 2017], the external attention [Guo *et al.*, 2022], and channel attention [Hu *et al.*, 2018]. Inspired by the recent attention-based work [Hu *et al.*, 2018] that proposes to squeeze the global spatial information into a channel descriptor, we commence by using

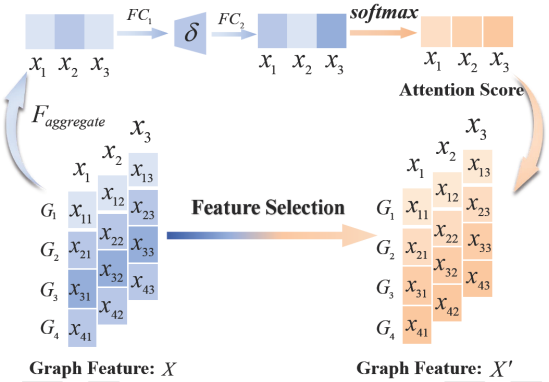


Figure 2: One Illustrative Instance of the Channel Attention Mechanism for Feature Selection.

the *SumPooling* operation to aggregate the information of all graphs and squeeze them into a feature channel. Given the feature matrix $\mathbf{X}_{(\cdot)} \in \mathbb{R}^{N \times L}$ of all graphs in $\mathbf{G}$, the aggregation information $\mathbf{h} \in \mathbb{R}^{1 \times L}$ can be calculated as

$$h_l = F_{\mathrm{aggregate}}(\boldsymbol{x_l}) = \frac{1}{N} \sum_{i=1}^{N} x_{l,i}, \quad (9)$$

where h_l is the l-th element of $\mathbf{h}$. With $\mathbf{h}$ to hand, we use three fully connected linear layers associated with the non-linear activation function to calculate the attention scores. Specifically, we use $W_1 \in \mathbb{R}^{L \times C}$ to denote the weight of the first fully connected layer and $W_2 \in \mathbb{R}^{C \times L}$ to represent the weight of the second dense layer, where C denotes the hidden feature dimension.

The resulting attention-based scoring matrix $\alpha \in \mathbb{R}^{N \times L}$ for the feature matrix $\mathbf{X}_{(\cdot)} \in \mathbb{R}^{N \times L}$ can be computed as

$$\alpha = \mathrm{softmax}(W_2 \sigma(W_1 \mathbf{h})), \quad (10)$$

where σ is the ReLU function, and α is the attention score.

With attention-based scoring matrix α that encapsulates adaptive weights for the different features of each graph, the

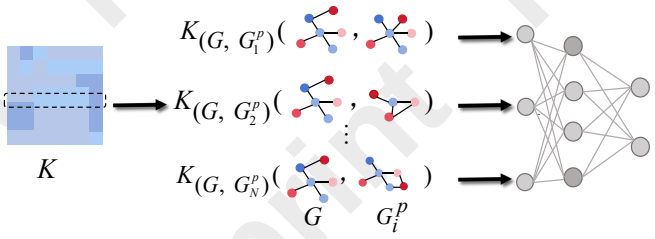


Figure 3: One Illustrative Instance of the Kernel-based Embedding.

substructure-based feature matrix $\mathbf{X}_{(\cdot)}$ can be updated as the weighted substructure-based feature matrix $\mathbf{X'}_{(\cdot)}$ by multiplying the attention scores, i.e.,

$$\mathbf{X'}_{(\cdot)} = \alpha \otimes \mathbf{X}_{(\cdot)}, \tag{11}$$

where $\otimes$ refers to the feature-wise multiplication. An instance of the attention mechanism for feature selection is shown in Figure2. In summary, the attention mechanism can assign substructures of each graph with different weights according to their importance.

The Kernel-based Graph Embedding Vectors

Based on the definition provided in [Shervashidze *et al.*, 2011], any R-convolution graph kernel can be computed as the dot product between the substructure-based feature vectors of pairwise graphs. For an instance, the WLSK kernel K_{WL} defined by Eq.(4) between a pair of graphs G_p and G_q can be rewritten as

$$K_{\mathrm{WL}}(G_p, G_q) = \langle \varphi_{\mathrm{WL}}(G_p), \varphi_{\mathrm{WL}}(G_q) \rangle, \tag{12}$$

where each $\varphi_{\mathrm{WL}}(G_p)$ is the substructure-based feature vector defined by Eq.(6). With the attention score computed by Eq.(11) to hand, we can compute the kernel value between weighted substructures as

$$K_{\mathrm{WL}}(G_p, G_q) = \langle \alpha \otimes \varphi_{\mathrm{WL}}(G_p), \alpha \otimes \varphi_{\mathrm{WL}}(G_q) \rangle. \tag{13}$$

Thus, the attention-based kernel matrix can be computed. Specifically, [Bunke and Riesen, 2008] has proposed a (dis)similarity graph embedding method that can embed or convert each graph structure into a vector, so that any standard machine learning and pattern recognition for vectors can be directly employed. In this paper, we propose to employ all sample graphs as the prototype graphs and use the graph kernel as the means of the similarity between each sample graph G and each prototype graph $G_q^p \in \mathbf{G}$ (including G itself). Thus, the kernel-based embedding vector $\phi_{(\cdot)}(G)$ of G can be defined as

$$\phi_{(\cdot)}(G) = [K_{(\cdot)}(G, G_1^p), \ldots, K_{(\cdot)}(G, G_q^p), \ldots, K_{(\cdot)}(G, G_N^p)], \tag{14}$$

where each element $K_{(\cdot)}(G, G_q^p)$ represents the kernel value between G and the prototype graph G_q^p, and $(\cdot)$ indicates that Eq.(14) can be computed with either the WLSK kernel or the SPGK kernel. Clearly, if $G \in \mathbf{G}$ is the j-th sample graph $G_j \in \mathbf{G}$ (i.e., $G = G_j$), the kernel-based embedding vector $\phi_{(\cdot)}(G)$ is essentially the j-th row of the kernel matrix $\mathbf{K}_{(\cdot)}$. As a result, the kernel matrix $\mathbf{K}_{(\cdot)}$ can be theoretically seen as the kernel-based embedding vectors over all graphs from $\mathbf{G}$.

An instance of the kernel-based graph embedding is shown in Figure 3, where G_i^p denotes the i-th prototype graph.

Since the ***attention-based weights*** for computing the kernel matrix and the ***trainable parameter matrix*** for the classifier can be adaptively updated when the loss is backpropagated, the computational framework of the proposed AKBR model can naturally provide an end-to-end learning mechanism.

3.3 Complexity Analysis

With $|V|$ and $|E|$ as the average number of nodes and edges, N is the number of graphs, T_{max} is the number of iterations in WLSK, d is the largest feature dimension. For the SPGK, we can extract all shortest paths in $O(|V|^2 \log |V| + |V||E|)$ time by using Johnson's algorithm. Since $0 < |E| \leq |V|(|V|-1)/2$, this approach only reaches cubic complexity for fully connected graphs. For WLSK, the time complexities of computing explicit feature vector and feature attention are $O(T_{max}|E|)$ and $O(Nd + d^2)$. Moreover, the time complexity of the classifier is $O(N^2 d)$. Thus, the total time complexities of AKBR(SP) and AKBR(WL) are $O(N(|V|^2 \log |V| + |V||E|) + N^2 d)$ and $O(T_{max}|E| + d^2 + N^2 d)$.

Datasets	# graphs	Mean # Node	# classes
MUTAG	188	17.93	2
PTC(MR)	344	25.56	2
PROTEINS	1113	39.06	2
D&D	1178	284.30	2
FRANKENSTEIN	4337	16.90	2
IMDB-B	1000	19.77	2
IMDB-M	1500	13.00	3
Shock	150	13.16	10
OGBG-MOLBACE	1513	34.10	2
OGBG-MOLBBBP	2039	24.06	2

Table 1: Information of the graph datasets

4 Experiments

In this section, we evaluate the performance of the proposed AKBR model against state-of-the-art graph kernels and deep learning methods. We use ten standard graph datasets extracted from bioinformatics (Bio), social networks (SN), and computer vision (CV). The OGBG-MOLBACE and OGBG-MOLBBBP datasets are selected from Open Graph Benchmark [Hu *et al.*, 2020]. The Shock dataset can be obtained from [Siddiqi *et al.*, 1999]. Other datasets from bioinformatics and social networks can be directly downloaded from [Morris *et al.*, 2020]. We provide the graph number and the average graph size of each dataset in Table 1. Our code is publicly available[1].

4.1 Comparisons with Graph Kernels

Experimental Settings

We compare the performance of the proposed AKBR model with several state-of-the-art graph kernels for graph classification tasks as Table 2: the Graphlet Count Graph Kernel (GCGK) with graphlet of size 3 [Shervashidze *et al.*, 2009], the Random Walk Graph Kernel (RWGK) [Gärtner *et*

[1]https://github.com/Sophia0830BNU/AKBR

Datasets	MUTAG	PTC(MR)	PROTEINS	IMDB-B	IMDB-M	Shock
GCGK3	82.04±0.39	55.41±0.59	71.67±0.55	65.87±0.98	45.42±0.87	26.93±0.63
RWGK	80.77±0.72	55.91±0.37	74.20±0.40	67.94±0.77	46.72±0.30	2.31±1.13
WL-OA	84.5±1.70	63.6±1.5	76.4±0.4	–	–	–
GAWL	87.3±6.3	–	74.7±3.0	74.5±4.1	51.7±5.2	–
HTAK	–	–	–	72.9±0.2	50.2±0.2	–
DHGAK	–	66.6±7.7	76.6±4.3	75.3±2.7	52.1±2.4	–
SPGK	83.38±0.81	55.52±0.46	75.10±0.50	71.26±1.04	51.33±0.57	37.88±0.93
CORE SP	88.29±1.55	59.06±0.93	–	72.62±0.59	49.43±0.42	–
AKBR(SP)	87.81±1.15	**65.83±1.27**	**75.40±2.10**	**75.19±0.47**	**52.61±0.44**	**41.22±3.98**
WLSK	82.88±0.57	58.26±0.47	73.52±0.43	71.88±0.77	49.50±0.49	36.40±1.00
CORE WL	87.47±1.08	59.43±1.20	–	74.02±0.42	51.35±0.48	–
AKBR(WL)	**89.47±0.56**	**68.35±1.08**	**77.97±0.43**	**75.35±0.57**	**52.06±0.49**	**44.43±3.01**

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

al., 2003], the Shortest Path Graph Kernel (SPGK) [Borgwardt and Kriegel, 2005], the Shortest Path Kernel based on Core Variants (CORE SP) [Nikolentzos *et al.*, 2018a], the Weisfeiler-Lehman Subtree Kernel (WLSK) [Shervashidze *et al.*, 2011], the WLSK kernel associated with Core Variants (CORE WL) [Nikolentzos *et al.*, 2018b], the Valid Optimal Assignment Kernel (WL-OA) [Kriege *et al.*, 2016], the Graph Alignment Kernels using Weisfeiler and Leman Hierarchies (GAWL) [Nikolentzos and Vazirgiannis, 2023], a new graph kernel that aligns vertices transitively between graph pairs (HTAK) [Bai *et al.*, 2022a] and the Deep Hierarchical Graph Alignment Kernels (DHGAK) leveraging natural language embeddings [Tang *et al.*, 2024]. We perform a 10-fold cross-validation and repeat the experiments ten times, and the average accuracy is reported in Table 2. Since some methods are not evaluated by the original paper on some datasets, we do not provide these results. We use the AKBR(SP) and AKBR(WL) to represent the AKBR model based on the SPGK and the WLSK.

Experimental Results and Analysis

Compared to the classical graph kernels, the family of the proposed AKBR models achieves highly competitive accuracies in Table 2. Specifically, we observe that the family of the proposed AKBR(WL) can outperform state-of-the-art graph kernels on all datasets. On the other hand, the proposed AKBR(SP) also significantly outperforms the original SPGK and Core SP kernels. These observations demonstrate that the theoretical advantages of the proposed AKBR model, i.e., adaptively identifying the importance of different substructures and computing the adaptive kernel matrix through an end-to-end learning framework can tremendously improve the classification performance.

4.2 Comparisons with Deep Learning

Experimental Settings

We further compare the family of our proposed AKBR models with some state-of-the-art graph deep learning methods as Table 4, including the Deep Graph Convolution Neural Network (DGCNN) [Zhang *et al.*, 2018], the Diffusion Convolution Neural Network (DCNN) [Atwood and Towsley, 2016], the PATCHY-SAN based Graph Convolution Neural Network (PATCHY-SAN) [Niepert *et al.*, 2016], the Deep Graphlet Kernel (DGK) [Yanardag and Vishwanathan, 2015], the Random Walk Graph Neural Networks (RWNN) [Nikolentzos and Vazirgiannis, 2020], the Graph Isomorphism Network (GIN) [Xu *et al.*, 2019], the WL-based GNNs (3WL-GNN) [Morris *et al.*, 2019], an instantiation of generalized message-passing framework (GraphSNN) [Wijesinghe and Wang, 2022], the GNNs with positional encoding (GatedGCN-LSPE) [Dwivedi *et al.*, 2022], the Graph Convolution Network (GCN) [Kipf and Welling, 2017], the Union Subgraph Neural Networks (UnionSNN) [Xu *et al.*, 2024] and the the masked-attention ML-model using softmax kernel (GKAT) [Choromanski *et al.*, 2022]. For these baseline methods, they are also evaluated using the same 10-fold cross-validation strategy as ours, thus we directly report the results from the original papers. We also evaluate the performance of the proposed AKBR method on the OGBG-MOLBACE and OGBG-MOLBBBP datasets. The results are reported in Table 3.

Model	OGBG-MOLBACE	OGBG-MOLBBBP
GraphSAGE	77.41±1.19	60.78±2.43
GraphSNN	–	62.84±0.36
UnionSNN	–	68.28±1.47
GIN	76.41±2.68	69.88±1.70
GCN	79.15±1.44	68.87±1.51
AKBR(WL)	**79.45±0.53**	**70.39±1.45**

Table 3: ROC-AUC score(± standard deviation) of the different approaches on OGBG-MOLBACE and OGBG-MOLBBBP datasets.

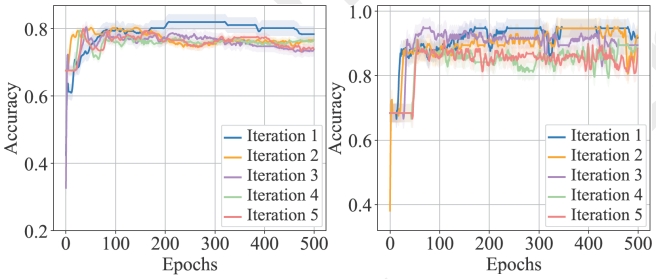
Experimental Results and Analysis

Table 4 demonstrates that the proposed AKBR(WL) model consistently outperforms alternative graph deep learning methods across seven TUDatasets. Furthermore, we also evaluate the proposed AKBR(WL) method on OGBG-MOLBACE and OGBG-MOLBBBP datasets. The results are shown in Table 3. We can observe that the proposed AKBR(WL) model achieves the highest ROC-AUC scores on the ogbg-molbace and ogbg-molbbbp datasets, reinforcing its effectiveness. In fact, similar to our methods, all these alternative graph deep learning methods can also provide an end-to-end learning framework, but typically stack more lay-

Datasets	MUTAG	PTC(MR)	PROTEINS	IMDB-B	IMDB-M	DD	FRANK
DGCNN	85.83±1.66	58.59±2.47	75.54±0.94	70.03±0.86	47.83±0.85	79.37±0.94	63.44±0.65
DCNN	66.98	56.60	61.29±1.60	49.06±1.37	46.72±0.30	58.09±0.53	–
PATCHY-SAN	88.95±4.37	62.29±5.68	75.00±2.51	71.00±2.29	45.23±2.84	76.27±2.64	–
DGK	82.66±1.45	60.08±2.55	71.68±0.50	66.96±0.56	44.55±0.52	78.50±0.22	–
3WL-GNN	84.06±6.62	60.9	60.18±6.35	74.2	49.5	74.84±2.63	58.68±1.93
RWNN	88.1±4.8	–	74.7±3.3	70.6±4.4	48.8±2.9	77.6±4.7	–
GraphSNN	84.04±4.09	61.63±2.8	71.78±4.11	74.81±3.5	–	76.03±2.59	67.17±2.25
GatedGCN-LSPE	88.33±3.88	–	73.94±2.72	70.03±5.15	46.47±4.00	76.74±2.04	67.74±2.65
GIN	84.7±6.7	64.29±1.26	74.3±3.3	71.23±3.9	48.53±3.3	75.3±2.9	66.50±2.37
UnionSNN	87.31±5.29	–	75.02±2.50	–	–	77.00±2.37	67.83±1.99
GKAT	–	–	75.80±3.70	71.40±2.60	47.5±4.5	78.6±3.4	–
AKBR(WL)	**89.47±0.56**	**68.35±1.08**	**77.97±0.43**	**75.35±0.57**	**52.06±0.49**	**80.02±0.42**	**71.25±0.23**

Table 4: Classification accuracy (in %±standard error) comparisons with deep learning methods.

ers than our model does. However, the proposed method still has better classification performance than these graph deep learning methods, again demonstrating the effectiveness of the kernel-based framework.



(a) PROTEINS (b) MUTAG

Figure 4: The evaluation for AKBR(WL) of iterations 1 to 5.

4.3 In-depth Discussions of the AKBR Model

To better analyze the advantages of the proposed AKBR model beyond standard benchmarks, we conduct an in-depth discussion focusing on its internal mechanisms and performance sensitivity.

Method	Iter	MUTAG	PROTEINS	DD
WL	1	40	300	253231
	2	214	21262	586248
	3	786	56938	920338
	4	1983	94878	1254664
	5	3749	133531	OOM
	6	5916	172457	OOM
	7	8319	211521	OOM
	8	10830	250662	OOM
	9	13390	289842	OOM
	10	15969	329045	OOM
SP	–	15	64	84

Table 5: Comparison of substructure counts for WLSK and SPGK.

Q: How sensitive is the model to the number of iterations? We explore how the classification accuracies of the proposed AKBR(WL) vary with different iterations ranging from 1 to 5 on the PROTEINS and MUTAG datasets, and the results are shown in Figure 4. Each model has been trained for 500 epochs on the first fold. We report curves with standard error computed on 40 runs for each iteration. Specifically, we find that AKBR(WL) with iteration 1 has the optimal performance. This is due to the fact that as the number of iterations increases, the number of substructures grows exponentially, resulting in a large amount of redundant information. To illustrate this conclusion more intuitively, we count the number of substructures generated by AKBR(WL) across iterations 1 to 10 as shown in Table 5.

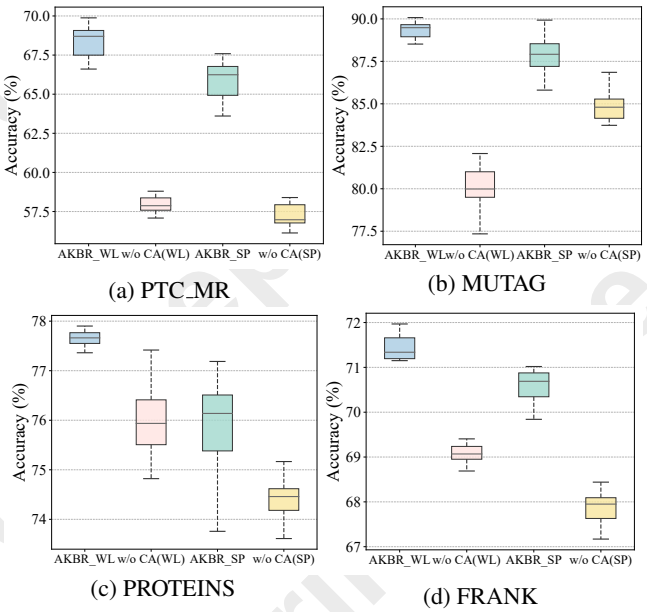
Model	MUTAG	PROTEINS	IMDB-B
AKBR_SA	86.10±2.18	72.04±1.03	70.93±1.06
AKBR_MHSA	85.85±2.24	70.44±1.32	71.32±1.30
AKBR_CA	**89.47±0.56**	**77.97±0.43**	**75.35±0.57**

Table 6: Ablation study on different attention mechanisms.

Q: How does the Channel Attention affect the performance of the proposed model? We first conduct the ablation study of the AKBR with or without channel attention on four representative datasets. Figure 5 shows the accuracy and standard deviation across ten runs of ten-fold cross-validation. When the Channel Attention is removed, the accuracy of the AKBR significantly decreases, demonstrating the importance of channel attention.

Furthermore, to more intuitively demonstrate the effectiveness of the channel attention mechanism, we incorporate Self-Attention and Multi-Head Self-Attention [Vaswani *et al.*, 2017] (with the number of heads set to 4) to the experiment, as shown in Table 6. To more intuitively demonstrate the effect of the attention mechanism, we also visualize the top four substructures with the highest attention scores in Figure 6. These substructures are representative of graph labels, indicating that our proposed channel attention effectively selects critical features, thereby enhancing the classification performance of the model.

Moreover, Table 5 shows that as the number of iterations increases, the number of substructures grows exponentially. Consequently, the weight assigned to each substructure in-

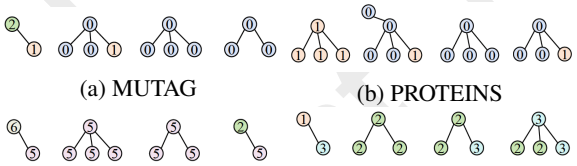


(a) PTC_MR (b) MUTAG

(c) PROTEINS (d) FRANK

Figure 5: The ablation study of AKBR with or without channel attention. w/o CA(WL) denotes AKBR(WL) without channel attention, w/o CA(SP) denotes AKBR(SP) without channel attention.

variant becomes smaller. This further explains why the performance of AKBR(WL) declines with additional iterations.



(a) MUTAG (b) PROTEINS

(c) PTC_MR (d) FRANK

Figure 6: The top four significant substructures for four datasets.

Q: Whether our proposed model can integrate other R-convolution graph kernels? Our proposed method is compatible with any R-convolution graph kernel. We choose the WLSK and SPGK because their associated subtree and shortest path substructures can be efficiently extracted and are known to represent meaningful structural information. To verify that our method is indeed applicable to other R-convolution graph kernels, we evaluate the AKBR model with the Random Walk Graph Kernel (RWGK), i.e., the AKBR (RW). As shown in Table 7, AKBR(RW) also achieves superior performance compared to the original RWGK, demonstrating that the proposed AKBR framework can be seamlessly integrated with various R-convolution kernels and consistently enhance their effectiveness.

Q: Whether the number of prototype graphs impacts the performance of the model? We conduct an experiment to evaluate the impact of the number of prototype graphs. The experimental results are shown as 8, the performance remains stable as the number of prototype graphs increases from 200 to 800 across all datasets. Note that, we select the graph with the highest Shannon entropy as the prototype graph [Bai *et al.*, 2013]. For example, on the DD dataset, the accuracy varies from 74.12% to 74.57%. These minimal variations indicate that the proposed method is robust to changes in the number of prototype graphs and maintains consistent performance, demonstrating strong generalization capability.

Method	MUTAG	PROTEINS	IMDB-B
RWGK	80.77±0.72	74.20±0.40	67.94±0.77
AKBR(RW)	**86.33±1.91**	**77.04±0.53**	**75.64±0.48**

Table 7: Performance comparison with Random Walk Graph Kernel.

Number of prototypes	PROTEINS	DD	FRANK
200	75.92 ± 0.67	74.12 ± 0.93	70.40 ± 0.34
400	75.90 ± 0.43	74.48 ± 0.78	70.49 ± 0.44
600	76.14 ± 0.43	74.57 ± 0.63	70.91 ± 0.30
800	75.94 ± 0.29	74.27 ± 0.37	70.51 ± 0.43

Table 8: Impact of the prototype graph number.

Q: Whether the AKBR(s) deliver superior efficiency compared with existing deep learning methods? We conduct an experiment to compare the total time cost of AKBR(WL) with that of deep learning methods. The results are shown in Table 9. For large-scale datasets, e.g., DD, our proposed AKBR model runs over 6.17 times faster than UnionSNN. This is because AKBR does not require a large number of network layers, further proving that our proposed model structure is simple yet effective.

Model	PROTEINS	DD	FRANK
3WL-GNN	6h3m36s	75h15m15s	19h18m36s
GraphSNN	4h3m	30h27m	5h45m36s
GatedGCN-LSPE	1h19m48s	3h39m	2h33m
GIN	31m48s	1h48m36s	2h0m36s
UnionSNN	1h18m36s	3h36m36s	2h27m36s
AKBR(WL)	**42s**	**35m4s**	**1m3s**

Table 9: Time cost for a single run with 10-fold-CV.

5 Conclusion

In this paper, we have proposed a novel AKBR model that can extract more effective substructure-based features and adaptively compute the kernel matrix for graph classification through an end-to-end learning framework. Thus, the proposed AKBR model can significantly address the shortcomings arising in existing R-convolution graph kernels. Experimental results show that our proposed AKBR model outperforms the existing state-of-the-art graph kernels and graph deep learning methods. In future work, we plan to extend our framework to incorporate a broader range of graph kernels, such as quantum walk-based kernels [Bai *et al.*, 2025; Bai *et al.*, 2024; Bai *et al.*, 2023], further enhancing its applicability and effectiveness. In addition, we will explore the hybrid graph kernels, to better capture complementary structural information from diverse perspectives.

Acknowledgements

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

[Atwood and Towsley, 2016] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Proceedings of NeurIPS*, volume 29, 2016.

[Aziz *et al.*, 2020] Furqan Aziz, Afan Ullah, and Faiza Shah. Feature selection and learning for graphlet kernel. *Pattern Recognition Letters*, 136:63–70, 2020.

[Bai *et al.*, 2013] Lu Bai, Edwin R Hancock, and Lin Han. A graph embedding method using the jensen-shannon divergence. In *Proceedings of CAIP*, pages 102–109, 2013.

[Bai *et al.*, 2020a] Lu Bai, Lixin Cui, Zhihong Zhang, Lixiang Xu, Yue Wang, and Edwin R Hancock. Entropic dynamic time warping kernels for co-evolving financial time series analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 34(4):1808–1822, 2020.

[Bai *et al.*, 2020b] 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. Springer, 2020.

[Bai *et al.*, 2022a] Lu Bai, Lixin Cui, and Edwin R. Hancock. A hierarchical transitive-aligned graph kernel for un-attributed graphs. In *Proceedings of ICML*, 2022.

[Bai *et al.*, 2022b] Lu Bai, Lixin Cui, Yuhang Jiao, Luca Rossi, and Edwin R. Hancock. Learning backtrackless aligned-spatial graph convolutional networks for graph classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(2):783–798, 2022.

[Bai *et al.*, 2023] Lu Bai, Yuhang Jiao, Lixin Cui, Luca Rossi, Yue Wang, Philip S. Yu, and Edwin R. Hancock. Learning graph convolutional networks based on quantum vertex information propagation. *IEEE Trans. Knowl. Data Eng.*, 35(2):1747–1760, 2023.

[Bai *et al.*, 2024] Lu Bai, Lixin Cui, Yue Wang, Ming Li, Jing Li, Philip S. Yu, and Edwin R. Hancock. HAQJSK: hierarchical-aligned quantum jensen-shannon kernels for graph classification. *IEEE Trans. Knowl. Data Eng.*, 36(11):6370–6384, 2024.

[Bai *et al.*, 2025] Lu Bai, Lixin Cui, Ming Li, Peng Ren, Yue Wang, Lichi Zhang, Philip S. Yu, and Edwin R. Hancock. AEGK: aligned entropic graph kernels through continuous-time quantum walks. *IEEE Trans. Knowl. Data Eng.*, 37(3):1064–1078, 2025.

[Borgwardt and Kriegel, 2005] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of ICDM*, pages 8–pp, 2005.

[Bruna *et al.*, 2014] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *Proceedings of ICLR*, 2014.

[Bunke and Riesen, 2008] Horst Bunke and Kaspar Riesen. Graph classification based on dissimilarity space embedding. In *Proceedings of SSPR*, pages 996–1007, 2008.

[Chang and Lin, 2011] C.-C Chang and C.-J. Lin. Libsvm: A library for support vector machines. *Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2011.

[Choromanski *et al.*, 2022] Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamás Sarlós, Adrian Weller, and Thomas Weingarten. From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In *Proceedings of ICML*, 2022.

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

[Cui *et al.*, 2024] Lixin Cui, Lu Bai, Xiao Bai, Yue Wang, and Edwin R. Hancock. Learning aligned vertex convolutional networks for graph classification. *IEEE Trans. Neural Networks Learn. Syst.*, 35(4):4423–4437, 2024.

[Du *et al.*, 2019] Simon S. Du, Kangcheng Hou, Ruslan Salakhutdinov, Barnabás Póczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Proceedings of NeurIPS*, pages 5724–5734, 2019.

[Dwivedi *et al.*, 2022] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *Proceedings of ICLR*. OpenReview.net, 2022.

[Feng *et al.*, 2022] Aosong Feng, Chenyu You, Shiqiang Wang, and Leandros Tassiulas. Kergnns: Interpretable graph neural networks with graph kernels. In *Proceedings of AAAI*, pages 6614–6622. AAAI Press, 2022.

[Floyd, 1962] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[Gärtner *et al.*, 2003] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of COLT*, pages 129–143, 2003.

[Gasteiger *et al.*, 2021] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Proceedings of NeurIPS*, 34:6790–6802, 2021.

[Guo *et al.*, 2022] Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu, and Shi-Min Hu. Beyond self-attention: External attention using two linear layers for visual tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5436–5447, 2022.

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

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of CVPR*, pages 7132–7141, 2018.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of NeurIPS*, 2020.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*, 2017.

[Kriege and Mutzel, 2012] Nils M. Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of ICML*, 2012.

[Kriege *et al.*, 2016] Nils M Kriege, Pierre-Louis Giscard, and Richard Wilson. On valid optimal assignment kernels and applications to graph classification. In *Proceedings of NeurIPS*, volume 29, 2016.

[Morris *et al.*, 2019] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order

graph neural networks. In *Proceedings of AAAI*, pages 4602–4609, 2019.

[Morris *et al.*, 2020] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[Niepert *et al.*, 2016] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of ICML*, pages 2014–2023, 2016.

[Nikolentzos and Vazirgiannis, 2018] Giannis Nikolentzos and Michalis Vazirgiannis. Message passing graph kernels. *arXiv preprint arXiv:1808.02510*, 2018.

[Nikolentzos and Vazirgiannis, 2020] Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Proceedings of NeurIPS*, 2020.

[Nikolentzos and Vazirgiannis, 2023] Giannis Nikolentzos and Michalis Vazirgiannis. Graph alignment kernels using weisfeiler and leman hierarchies. In *Proceedings of AISTATS*, pages 2019–2034. PMLR, 2023.

[Nikolentzos *et al.*, 2018a] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. A degeneracy framework for graph similarity. In *Proceedings of IJCAI*, pages 2595–2601, 2018.

[Nikolentzos *et al.*, 2018b] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. A degeneracy framework for graph similarity. In *Proceedings of IJCAI*, pages 2595–2601, 2018.

[Shervashidze *et al.*, 2009] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, 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. *Journal of Machine Learning Research*, 12(9), 2011.

[Siddiqi *et al.*, 1999] Kaleem Siddiqi, Ali Shokoufandeh, Sven J Dickinson, and Steven W Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35:13–32, 1999.

[Tang *et al.*, 2024] Shuhao Tang, Hao Tian, Xiaofeng Cao, and Wei Ye. Deep hierarchical graph alignment kernels. In *Proceedings of IJCAI*, pages 4964–4972. ijcai.org, 2024.

[Togninalli *et al.*, 2019] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *Proceedings of NeurIPS*, 32, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NeurIPS*, volume 30, 2017.

[Weisfeiler and Lehman, 1968] B. Weisfeiler and A.A. Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, Ser.2(9), 1968.

[Wijesinghe and Wang, 2022] Asiri Wijesinghe and Qing Wang. A new perspective on" how graph neural networks go beyond weisfeiler-lehman?". In *Proceedings od ICLR*, 2022.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of ICLR*. OpenReview.net, 2019.

[Xu *et al.*, 2024] Jiaxing Xu, Aihu Zhang, Qingtian Bian, Vijay Prakash Dwivedi, and Yiping Ke. Union subgraph neural networks. In *Proceedings of AAAI*, pages 16173–16183, 2024.

[Yanardag and Vishwanathan, 2015] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of SIGKDD*, pages 1365–1374, 2015.

[Zambon *et al.*, 2018] Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Concept drift and anomaly detection in graph streams. *IEEE transactions on neural networks and learning systems*, 29(11):5592–5605, 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*, 2018.