Article:
Bai, Lu, Jiao, Yuhang, Cui, Lixin et al. (4 more authors) (2023) Learning Graph Convolutional Networks based on Quantum Vertex Information Propagation. IEEE Transactions on Knowledge and Data Engineering. pp. 1747-1760. ISSN: 1041-4347

https://doi.org/10.1109/TKDE.2021.3106804

Learning Graph Convolutional Networks based on Quantum Vertex Information Propagation

Lu Bai[1,2,3], Yuhang Jiao[1], Lixin Cui[1,2,3*], Luca Rossi[4], Yue Wang[1],
Philip S. Yu[5], *IEEE Fellow* and Edwin R. Hancock[6], *IEEE Fellow*

***Abstract*—This paper proposes a new Quantum Spatial Graph Convolutional Neural Network (QSGCNN) model that can directly learn a classification function for graphs of arbitrary sizes. Unlike state-of-the-art Graph Convolutional Neural Network (GCNN) models, the proposed QSGCNN model incorporates the process of identifying transitive aligned vertices between graphs and transforms arbitrary sized graphs into fixed-sized aligned vertex grid structures. In order to learn representative graph characteristics, a new quantum spatial graph convolution is proposed and employed to extract multi-scale vertex features, in terms of quantum information propagation between grid vertices of each graph. Since the quantum spatial convolution preserves the grid structures of the input vertices (i.e., the convolution layer does not alter the original spatial position of vertices), the proposed QSGCNN model allows to directly employ the traditional convolutional neural network architecture to further learn from the global graph topology, providing an end-to-end deep learning architecture that integrates the graph representation and learning in the quantum spatial graph convolution layer and the traditional convolutional layer for graph classifications. We indicate the effectiveness of the proposed QSGCNN model in relation to existing state-of-the-art methods. Experiments on benchmark graph classification datasets demonstrate the effectiveness of the proposed QSGCNN model.**

***Index Terms*—Graph Neural Networks, Quantum Walks, Quantum Graph Convolution, Quantum Propagation**

I. Introduction

Graph-based representations have been widely employed to model and analyze data that lies on high-dimensional non-Euclidean domains and that is naturally described in terms of pairwise relationships between its parts [1]. Typical instances where data can be represented using graphs include a) classifying proteins or chemical compounds [2], b) recognizing objects from digital images [3], c) visualizing social networks [4]. A fundamental challenge arising in the analysis of real-world data represented as graphs is the lack of a clear and accurate way to represent discrete graph structures as numeric features that can be directly analyzed by standard machine learning techniques [5]. This paper aims to develop a new graph convolutional neural network using quantum vertex saliency, for the purpose of graph classification. Our method is based on identifying the transitive alignment information between vertices of all different graphs. That is, given three vertices v, w and x from three sample graphs, suppose v and x are aligned, and w and x are aligned, the proposed model can guarantee that v and w are also aligned. The alignment procedure not only provides a way of mapping each graph into a fixed-sized vertex grid structure, but also bridges the gap between the graph convolution layer and the traditional convolutional neural network layer.

Lu Bai, Yuhang Jiao, Lixin Cui (*Corresponding Author: cuilixin@cufe.edu.cn), and Yue Wang are with [1]Engineering Research Center of State Financial Security, Ministry of Education, Central University of Finance and Economics, Beijing, China, and [2]Research Center of Artificial Intelligence in Accounting, Central University of Finance and Economics, Beijing, China, [3]Research Center of Artificial Intelligence in Accounting, Central University of Finance and Economics, Beijing, China. Luca Rossi is with [4]Queen Mary University of London, London, UK. Philip S. Yu is with [5]Department of Computer Science, University of Illinois at Chicago, US. Edwin R. Hancock is with [6]Department of Computer Science, University of York, York, UK. This work is supported by the National Natural Science Foundation of China (Grant no. 61976235 and 61602535), Nature Science Research Project of Anhui province (1908085MF185), China's Post-doctoral Science Fund (2020M681989), and the program for innovation research in Central University of Finance and Economics.

A. Literature Review

There have been a large number of methods aimed at converting graph structures into numeric representations, thus providing a way of directly applying standard machine learning algorithm to problems of graph classification or clustering. Generally speaking, in the last three decades, most classical state-of-the-art approaches to the analysis of graph structures can be divided into two classes, namely 1) graph embedding methods and 2) graph kernels. The methods from the first class aim to represent graphs as vectors of permutation invariant features, so that one can directly employ standard vectorial machine learning algorithms [6]. All of the previous approaches are based on the computation of explicit embeddings into low dimensional vector spaces, which inevitably leads to the loss of structural information. Graph kernels, on the other hand, try to soften this limitation by (implicitly) mapping graphs to a high dimensional Hilbert space where the structural information is better preserved [7]. The majority of state-of the-art graph kernels are instances of the R-convolution kernel originally proposed by Haussler [8]. The main idea underpinning R-convolution kernels is that of decomposing graphs into substructures (e.g, walks, paths, subtrees, and subgraphs) and then to measure the similarity between a pair of input graphs in terms of the similarity between their constituent substructures. Representative R-convolution graph kernels include the Weisfeiler-Lehman subtree kernel [9], the subgraph matching kernel [10], the aligned subtree kernel [11], and the aligned subgraph kernel [12]. A common limitation shared by both graph embedding methods and kernels is that of ignoring information from multiple graphs. This is because graph embedding methods usually capture structural features of individual graphs, while graph kernels reflect structural characteristics for pairs of graphs.

Recently, deep learning networks have emerged as an effective way to extract highly meaningful statistical patterns in large-scale and high-dimensional data [13]. As evidenced by their recent successes in computer vision problems, convolutional neural networks (CNNs) [14] are one of the most popular class of deep learning architectures and many researchers have devoted their efforts to generalizing CNNs to the graph domain [15]. Unfortunately, applying CNNs for graphs in a straightforward way is not trivial, since these networks are designed to operate on regular grids [1] and the associated operations of convolution, pooling and weight-sharing cannot be easily extended to graphs.

To address the aforementioned problems, two popular strategies have been proposed and employed to extend convolutional neural networks to graph domains, i.e., the spectral and the spatial strategies. Specifically, approaches using the spectral strategy utilise the property of the convolution operator from the graph Fourier domain, and relate to the graph Laplacian [16]. By transforming the graph into the spectral domain through the Laplacian matrix eigenvectors, the filter operation is performed by multiplying the graph by a series of filter coefficients. Unfortunately, most spectral-based approaches demand the size of the graph structures to be the same and cannot be performed on graphs with different sizes and Fourier bases. As a result, approaches based on the spectral strategy are usually applied to vertex classification tasks. By contrast, methods based on the spatial strategy are not restricted to the same graph structure. These methods generalize the convolution operation to the spatial structure of a graph by propagating features between neighboring vertices [17]. For instance, Duvenaud et al. [18] have proposed a Neural Graph Fingerprint Network by propagating vertex features between their 1-layer neighbors to simulate the traditional circular fingerprint. Atwood and Towsley [19] have proposed a Diffusion Convolution Neural Network by propagating vertex features between neighbors of different layers rooted at a vertex. Although spatially based approaches can be directly applied to real-world graph classification problems, most existing methods have fairly poor performance. This is because these methods tend to directly sum up the extracted local-level vertex features from the convolution operation as global-level graph features through a SumPooling layer. It is then difficult to learn the topological information residing in a graph through these global features.

To overcome the shortcoming of the graph convolutional neural networks associated with SumPooling, unlike the works in [18] and [19], Nieper et al. [20] have developed a different graph convolutional neural network by constructing a fixed-sized local neighborhood for each vertex and re-ordering the vertices based on graph labeling methods and graph canonization tools. This procedure naturally forms a fixed-sized vertex grid structure for each graph, and the graph convolution operation can be performed by sliding a fixed-sized filter over spatially neighboring vertices. This operation is similar to that performed on images with standard convolutional neural networks. Zhang et al. [21] have developed a novel Deep Graph Convolutional Neural Network model that can preserve more vertex information and learn from the global graph topology. Specifically, this model utilizes a newly developed SortPooling layer, that can transform the extracted vertex features of unordered vertices from spatial graph convolution layers into a fixed-sized vertex grid structure. Then a traditional convolutional neural networks can be applied to the grid structures to further learn the graph topological information.

Although both methods of Nieper et al. [20] and Zhang et al. [21] outperform state-of-the-art graph convolutional neural network models and graph kernels on graph classification tasks, these approaches suffer from the drawback of ignoring structural correspondence information between graphs, or rely on simple but inaccurate heuristics to align the vertices of the graphs, i.e., they sort the vertex orders based on the local structure descriptor of each individual graph and ignore the vertex correspondence information between different graphs. As a result, both the methods cannot reflect the precise topological correspondence information for graph structures. These approaches also lead to significant information loss. This usually occurs when these approaches form the fixed-sized vertex grid structure and some vertices associated with lower ranking may be discarded. In summary, developing effective methods to preserve the structural information residing in graphs still remains a significant challenge.

B. Contribution

The aim of this paper is to overcome the shortcomings of the aforementioned methods by developing a new Quantum Spatial Graph Convolutional Neural Network (QSGCNN) model. The starting point of the new model is the identification of the transitive vertex alignment information between graphs. Specifically, the new model can employ the transitive alignment information to map different sized graphs into fixed-sized aligned representations, i.e., it can transform different graphs into fixed-sized aligned grid structures with consistent vertex orders. With the aligned grid structures of graphs to hand, a novel quantum spatial graph convolutional operation is developed to further extract multi-scale graph features from the grid structures. The aligned grid structure can precisely integrate the structural correspondence information and all the original vertex information will be mapped into the grid structure through the transitive alignment, i.e., the mapping process does not discard any vertex. The proposed graph convolutional operation associated with the aligned grid structures can not only bridge the gap between the spatial graph convolution layer and the traditional convolutional neural network layer, but also reduce the shortcomings of information loss and imprecise information representation arising in most state-of-the-art graph convolutional neural networks associated with SortPooling or SumPooling layers. The computational architecture of the proposed model is shown in Fig.1. Specifically, the main contributions of this work are threefold.

First, we introduce a framework for transitively aligning the vertices of a family of graphs in terms of vertex point matching. This framework can establish reliable vertex correspondence information between graphs, by gradually minimizing the inner-vertex-cluster sum of squares over the vertices of all graphs. We show that this framework can be further employed to map graphs of arbitrary sizes into fixed-sized

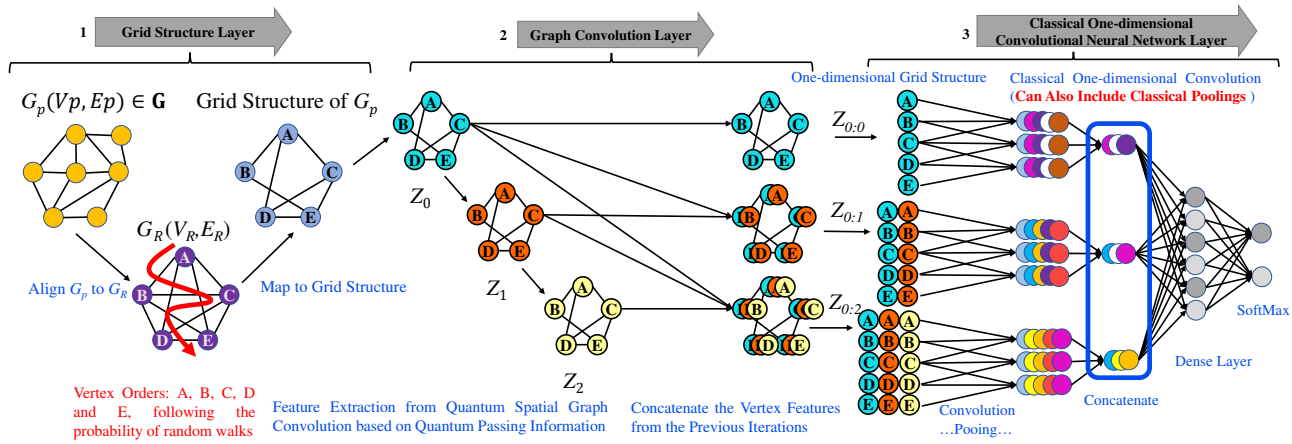


Fig. 1. **The architecture of the proposed QSGCNN model**. **(1)** An input graph $G_p(V_p, E_p) \in \mathbf{G}$ of arbitrary size is first aligned to the prototype graph $G_R(V_R, E_R)$, by identifying the structure correspondence information between the vertices of G_p and G_R. Then, the vertices of G_p aligned to the same vertex of G_R will be mapped into the same aligned vertex, where each aligned vertex follows the same vertex order of the corresponding vertex of G_R, i.e., these new aligned vertices follow the same vertex spatial positions of G_R. Here, the red curved arrow on the graph G_R indicates the predetermined spatial orders of its vertices. This process in turn forms a nature fixed-sized aligned vertex grid structure (see details in III-A), where a standard CNN can be directly performed. Since the above construction process will not discard any original vertex of G_p, the resulting aligned vertex grid structure can reduce the problem of information loss that arises in existing graph convolutional neural network models associated with the SortPolling operation. **(2)** The grid structure of G_p is passed through multiple quantum spatial graph convolution layers to extract multi-scale vertex features, where the vertex information is propagated between specified vertices following the average mixing matrix. **(3)** Since the graph convolution layers preserve the original vertex orders of the input grid structure, the concatenated vertex features through the graph convolution layers form a new vertex grid structure for G_p. This vertex grid structure is then passed to a traditional CNN layer to learn a classification function. Note, vertex features are visualized as different colors.

aligned vertex grid structures, integrating precise structural correspondence information and thus minimizing the loss of structural information. The resulting grid structures can bridge the gap between the spatial graph convolution layer and the traditional convolutional neural network layer.

Second, with the aligned vertex grid structures and their associated adjacency matrices to hand, we propose a novel quantum spatial graph convolution layer to extract multi-scale vertex features. Unlike the existing spatial graph convolution neural network models that propagate features between specified vertices through the vertex adjacency matrix (i.e., these models rely on the vertex visiting probability information of classical random walks [22]), the proposed graph convolution layer propagates the feature information between aligned grid vertices based on the vertex visiting information of continuous-time quantum walks [23]. In quantum information theory [24], the continuous-time quantum walk is the natural quantum analogue of the classical random walk [23], and has been widely employed to develop novel quantum algorithms in machine learning and data mining [25]. More specifically, in this work we employ the average mixing matrix to capture the visiting information of the quantum walks. The reasons for using the quantum walk is that it not only reduces the tottering effect arising in classical random walks, but it also reflects richer graph characteristics than classical random walks [23] (see details in Section II-A). We show that the new convolution layer not only overcomes the aforementioned information loss problem of popular graph convolutional neural networks associated with SortPooling or SumPooling layers, but also reduces the notorious tottering problem of existing graph kernels based on the Weisfeiler-Lehman algorithm [9] (see details in Section III-D) that may result in redundant information [26]. This in turn support the empirical evidence collected in our experimental validation. Moreover, since the proposed convolution layer does not alter the original spatial position of vertices, it also allows us to directly employ the traditional convolutional neural network to further learn from the global graph topology, providing an end-to-end deep learning architecture that integrates the graph representation and learning into both the quantum spatial graph convolution and the traditional convolutional layers for graph classification.

Third, we empirically evaluate the proposed Quantum Spatial Graph Convolutional Neural Network (QSGCNN). Experimental results on benchmark graph classification datasets demonstrate that our proposed QSGCNN significantly outperforms state-of-the-art graph kernels and deep graph convolutional network models for graph classifications.

II. Preliminary Concepts

A. Continuous-time Quantum Walks

One main objective of this work is to develop a new spatial graph convolution layer to extract multi-scale vertex features by gradually propagating information for each vertex to its neighboring vertices as well as the vertex itself. This usually requires connection information between each vertex and its neighboring vertices. Most existing methods employ the vertex adjacency matrix of each graph in the formulation of the information propagation framework [18], [19], [20], [21], i.e., these methods rely on the vertex visiting information of classical random walks. Recently, quantum algorithms have been used to develop novel approaches in machine learning and data mining [27], because of the richer structure than their classical counterparts. For instance, Melucci [28] has developed a relevance feedback algorithm based on the quantum probability subspace [29]. Fawaz et al. [30] have developed a novel strategy to train binary neural networks associated with quantum amplitude amplifications. In this work, in order to capture richer vertex features from the proposed graph convolutional layer, we employ the vertex information prop-

agation process of the continuous-time quantum walk, that is the quantum analogue of the classical random walk [23].

The main reason for relying on quantum walks is that, unlike classical random walks, whose state is described by a real-valued vector and where the evolution is governed by a doubly stochastic matrix, the state vector of the quantum walks is complex-valued and its evolution is governed by a time-varying unitary matrix. Thus, the quantum walk evolution is reversible, implying that it is non-ergodic and does not possess a limiting distribution. As a result, the behaviour of quantum walks is significantly different from their classical counterpart and possesses a number of important properties, e.g., it allows interference to take place. This interference, in turn, helps to reduce the tottering problem of random walks, as a quantum walkers backtracking on an edge does so with reversed phase. Furthermore, since the evolution of the quantum walk is not dominated by the low frequency components of the Laplacian spectrum, it has better ability to distinguish different graph structures. In Section III, we will show that the proposed graph convolutional layer associated with the continuous-time quantum walk can not only reduce the tottering problem arising in some state-of-the-art graph kernels and graph convolutional network models, but also better discriminate between different graphs.

In this subsection, we briefly review the concept of continuous-time quantum walks. Specifically, we use the average mixing matrix to capture the time-averaged behaviour of the quantum walk and to measure the quantum information being transmitted between the graph vertices. The continuous-time quantum walk is the quantum analogue of the continuous-time classical random walk [23], where the latter models a Markovian diffusion process over the vertices of a graph through the transitions between adjacent vertices. Let a sample graph be denoted as $G(V,E)$ with vertex set V and edge set E. Like the classical random walk, the state space of the quantum walk is the vertex set V. Using the Dirac notation, the basis state of the quantum walk being at vertex $u \in V$ is defined as $|u\rangle$, where $|.\rangle$ corresponds to an orthonormal vector in a $|V|$-dimensional complex-valued Hilbert space $\mathcal{H}$. Its state $|\psi(t)\rangle$ at time t is a complex linear combination of these orthonormal basis states $|u\rangle$, i.e.,

$$|\psi(t)\rangle = \sum_{u\in V} \alpha_u(t)\,|u\rangle\,, \tag{1}$$

where $\alpha_u(t) \in \mathbb{C}$ is the complex amplitude. Furthermore, $\alpha_u(t)\alpha_u^*(t)$ indicates the probability of the walker visiting vertex u at time t, where $\alpha_u^*(t)$ is the complex conjugate of $\alpha_u(t)$, $\sum_{u\in V}\alpha_u(t)\alpha_u^*(t)=1$, and $\alpha_u(t)\alpha_u^*(t)\in[0,1]$ for all $u\in V$ and $t\in\mathbb{R}^+$. Unlike the classical counterpart, the continuous-time quantum walk evolves based on the Schrödinger equation

$$\partial/\partial t\,|\psi_t\rangle = -i\mathcal{H}\,|\psi_t\rangle\,, \tag{2}$$

where $\mathcal{H}$ represents the system Hamiltonian and accounts for the total energy of the system. In this work, we use the adjacency matrix as the Hamiltonian. The behaviour of a quantum walk over the graph $G(V,E)$ at time t can be summarized using the mixing matrix [31]

$$Q_M(t) = U(t)\circ U(-t) = e^{i\mathcal{H}t}\circ e^{-i\mathcal{H}t}, \tag{3}$$

where the operation symbol $\circ$ represents the Schur-Hadamard product of $e^{i\mathcal{H}t}$ and $e^{-i\mathcal{H}t}$. Because U is unitary, $Q_M(t)$ is a doubly stochastic matrix and each entry $Q_M(t)_{uv}$ indicates the probability of the walk visiting vertex v at time t when the walk initially starts from vertex u. However, $Q_M(t)$ cannot converge, because $U(t)$ is also norm-preserving. To overcome this problem, we can enforce convergence by taking a time average. Specifically, we take the Cesàro mean and define the average mixing matrix as

$$Q = \lim_{T\to\infty}\int_0^T Q_M(t)dt, \tag{4}$$

where each entry $Q_{v_iv_j}$ of the average mixing matrix Q represents the average probability for a quantum walk to visit vertex v_j starting from vertex v_i, and Q is still a doubly stochastic matrix. Furthermore, Godsil [31] has indicated that the entries of Q are rational numbers. We can easily compute Q from the spectrum of the Hamiltonian. Specifically, let the adjacency matrix A of G be the Hamiltonian $\mathcal{H}$. Let $\lambda_1,\ldots,\lambda_{|V|}$ represent the $|V|$ distinct eigenvalues of H and $\mathbb{P}_j$ is the matrix representation of the orthogonal projection on the eigenspace associated with the λ_j, i.e., $\mathcal{H}=\sum_{j=1}^{|V|}\lambda_j\mathbb{P}_j$. Then, we can re-write the average mixing matrix Q as

$$Q = \sum_{j=1}^{|V|}\mathbb{P}_j\circ\mathbb{P}_j. \tag{5}$$

B. Transitive Alignment Between Vertices of Graphs

We introduce a new transitive vertex alignment method. To this end, we commence by identifying a family of prototype representations that reflect the main characteristics of the vectorial vertex representations over a set of graphs $\mathbf{G}$. Assume there are n vertices over all graphs in $\mathbf{G}$, and the associated K-dimensional vectorial representations of these vertices are $\mathbf{R}^K=(\mathrm{R}_1^K,\mathrm{R}_2^K,\ldots,\mathrm{R}_n^K)$, we use k-means [33] to identify M centroids over all representations in $\mathbf{R}^K$. Specifically, given M clusters $\Omega=(c_1,c_2,\ldots,c_M)$, the aim of k-means is to minimize the following objective function

$$\arg\min_{\Omega}\sum_{i=1}^{M}\sum_{\mathrm{R}_j^K\in c_i^K}\|\mathrm{R}_j^K-\mu_i^K\|^2, \tag{6}$$

where μ_i^K is the mean of the vectorial vertex representations belonging to the i-th cluster c_i. Since Eq.(6) minimizes the sum of the square Euclidean distances between the vertex points R_j^K and the centroid point of cluster c_i^K, the M centroid points $\{\mu_1^K,\cdots,\mu_i^K,\cdots,\mu_M^K\}$ can be seen as a family of K-dimensional **prototype representations** that encapsulate representative characteristics over all graphs in $\mathbf{G}$.

Let $\mathbf{G}=\{G_1,\cdots,G_p,\cdots,G_q,\cdots,G_N\}$ be a set of graphs. For each graph $G_p(V_p,E_p)\in\mathbf{G}$ and each vertex $v_i\in V_p$ associated with its K-dimensional vectorial representation $\mathrm{R}_{p;i}^K$, we commence by identifying the set of K-dimensional prototype representations as $\mathbf{PR}^K=\{\mu_1^K,\ldots,\mu_j^K,\ldots,\mu_M^K\}$

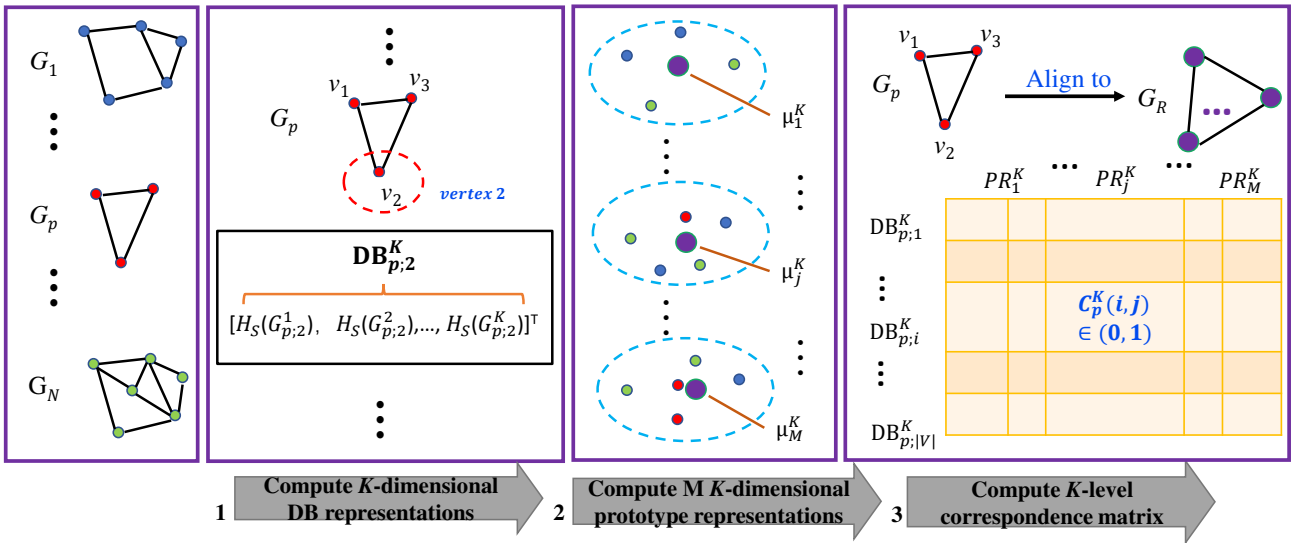


Fig. 2. **The procedure for computing the vertex correspondence matrix**. Given a set of graphs $\mathbf{G}$, for each sample graph $G_p \in \mathbf{G}$: **(1)** we compute the K-dimensional DB representation $\mathrm{DB}^K_{p;v}$ rooted at each vertex (e.g., vertex v_2 of G_p). We represent this as a K-dimensional vertex vector, where each element $H_s(G^K_{p;2})$ of $\mathrm{DB}^K_{p;v}$ represents the Shannon entropy of the K-layer expansion subgraph rooted at v_2 [32]. **(2)** We identify a family of K-dimensional prototype representations $\mathbf{PR}^K = \{\mu^K_1, \ldots, \mu^K_j, \ldots, \mu^K_M\}$ (shown in purple) by applying the k-means algorithm to the K-dimensional DB representations of the complete set of sample graphs, i.e., we construct M mean vectorial representations of M clusters through k-menas. **(3)** We align the K-dimensional DB representations to the K-dimensional prototype representations and compute a K-level correspondence matrix C^K_p. The correspondence matrix C^K_p records the correspondence information, where the element $C^K_p(i,j) = 1$ indicates a structural correspondence between the i-th vertex of G_p and the j-th vertex of G_R.

for the graph set $\mathbf{G}$. To establish a set of correspondences between the graph vertices, we align the vectorial vertex representations of each graph G_p to the family of prototype representations $\mathbf{PR}^K$. The alignment process is similar to that introduced in [11] for point matching in a pattern space. Specifically, we compute a K-level affinity matrix in terms of the Euclidean distances between the two sets of points

$$A^K_p(i,j) = \|\mathrm{R}^K_{p;i} - \mu^K_j\|_2. \tag{7}$$

where A^K_p is a $|V_p| \times M$ matrix, and each element $A^K_p(i,j)$ represents the distance between the vectorial representation $\mathrm{R}^K_{p;i}$ of $v_{\in}V_p$ and the j-prototype representation $\mu^K_j \in \mathbf{PR}^K$. If the value of $A^K_p(i,j)$ is the smallest in row i, we say that $\mathrm{R}^K_{p;i}$ is aligned to μ^K_j, i.e., the vertex v_i is aligned to the j-th prototype representation. Note that for each graph there may be two or more vertices aligned to the same prototype representation. We record the correspondence information using the K-level correspondence matrix $C^K_p \in \{0,1\}^{|V_p|\times M}$

$$C^K_p(i,j) = \begin{cases} 1 & \text{if } A^K_p(i,j) \text{ is the smallest in row } i \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

For a pair of graphs G_p and G_q, if their vertices v_p and v_q are aligned to the same prototype representation PR^K_j, we say that v_p and v_q are also aligned. Thus, we can identify the transitive alignment information between the vertices of all graphs in $\mathbf{G}$, by matching their vertices to a common set of reference points, i.e., the prototype representations.

To construct reliable correspondence information for the graphs, in this work we employ a depth-based (DB) representation [32] as the initial vectorial vertex representations (i.e.$\mathbf{R}^K$). This is because the DB representation of each vertex is computed by measuring the entropies on a family of k-layer expansion subgraphs rooted at the vertex, where the parameter k varies from 1 to K. It has been shown that such a K-dimensional DB representation can be viewed as ***a nested vertex representation*** that encapsulates a rich nested entropy-based information content flow from each local vertex to the global graph structure, as a function of depth. Fig.2 illustrates the process of computing the K-level correspondence matrix C^K_p associated with DB representations.

III. The Quantum Spatial Graph Convolutional Neural Network

In this section, we develop a new Quantum Spatial Graph Convolutional Neural Network (QSGCNN) model. The architecture of the proposed model has been shown in Fig.1. Specifically, the architecture is composed of three sequential stages, i.e., 1) the grid structure construction and input layer, 2) the quantum spatial graph convolution layer, and 3) the traditional convolutional neural network and Softmax layers. Specifically, the grid structure construction and input layer a) first maps graphs of arbitrary sizes into fixed-sized grid structures with consistent vertex orders, and b) inputs the grid structures into the proposed QSGCNN model. With the input graph grid structures to hand, the quantum spatial graph convolution layer further extracts multi-scale vertex features by propagating vertex feature information between the aligned grid vertices. Since the extracted vertex features from the graph convolution layer preserve the original vertex orders of the input grid structures, the traditional convolutional neural network and Softmax layer can read the extracted vertex features and predict the graph class.

A. Aligned Vertex Grid Structures of Graphs

In this subsection, we show how to map graphs of different sizes onto fixed-sized aligned vertex grid structures and associated corresponding fixed-sized aligned grid vertex adjacency matrices. For the set of graphs $\mathbf{G}$ defined earlier, suppose $G_p(V_p, E_p, A_p) \in \mathbf{G}$ is a sample graph, with V_p representing the vertex set, E_p representing the edge set, and A_p representing the vertex adjacency matrix. Suppose each vertex $v_p \in V_p$ is represented as a c-dimensional feature vector. Then the features of all the n vertices can be encoded using the $n \times c$ matrix X_p, i.e., $X_p \in \mathbb{R}^{n\times c}$. Note that the row of X_p follows the same vertex order of A_p. If the graphs in $\mathbf{G}$ are vertex attributed graphs, X_p can be the one-hot encoding matrix of the vertex labels. For unattributed graphs, we propose to use the vertex degree as the vertex label. Based on the transitive vertex alignment method introduced in Section II, for each graph $G_p \in \mathbf{G}$, we commence by computing the K-level vertex correspondence matrix C_p^K that records the correspondence information between the K-dimensional vectorial vertex representation of G_p and the K-dimensional prototype representations in $\mathbf{PR}^K = \{\mu_1^K, \ldots, \mu_j^K, \ldots, \mu_M^K\}$ of $\mathbf{G}$. The rows and columns of C_p^K are indexed by the vertices in V_p and the prototype representations in $\mathbf{PR}^K$, respectively. With C_p^K to hand, we compute the K-level aligned vertex feature matrix for G_p as

$$\widehat{X}_p^K = (C_p^K)^T X_p, \tag{9}$$

where $\widehat{X}_p^K$ is a $M \times c$ matrix and each row of $\widehat{X}_p^K$ represents the feature of a corresponding aligned vertex. Moreover, we also compute the associated K-level aligned vertex adjacency matrix for G_p as

$$\widehat{A}_p^K = (C_p^K)^T (A_p)(C_p^K), \tag{10}$$

where $\widehat{A}_p^K$ is a $M \times M$ matrix. With the correspondence matrix C_p^K to hand, $\widehat{X}_p^K$ and $\widehat{A}_p^K$ are computed from the original vertex feature matrix and adjacency matrix, respectively, by mapping the original feature and adjacency information of each vertex $v_p \in V_p$ to that of the new aligned vertices indexed by the corresponding prototypes in $\mathbf{PR}^K$. In other words $\widehat{X}_p^K$ and $\widehat{A}_p^K$ encapsulate the original feature and structural information of G_p. Note also that according to Eq. 8 each vertex $v_p \in V_p$ can be aligned to more than one prototype, and thus in general $\widehat{A}_p^K$ is a weighted adjacency matrix.

In order to construct the fixed-sized aligned grid structure for each graph $G_p \in \mathbf{G}$, we need to establish a consistent order for the vertices of each graph. Since the vertices of each graph are aligned to the same prototype representations, we determine the vertex orders by reordering the prototype representations. To this end, we compute a quasi graph $G_{\mathrm{R}}(V_{\mathrm{R}}, E_{\mathrm{R}})$ with each vertex $v_j \in V_{\mathrm{R}}$ representing the prototype $\mu_j^K \in \mathbf{PR}^K$ and each edge $(v_j, v_k) \in E_{\mathrm{R}}$ representing the similarity between $\mu_j^K \in \mathbf{PR}^K$ and $\mu_k^K \in \mathbf{PR}^K$. Specifically, we employ the well-known Gaussian kernel [1] (a widely used way of characterising attribute vector similarity) to compute the similarity between two vertices of G_R

$$s(\mu_j^K, \mu_k^K) = \exp(-\frac{\|\mu_j^K - \mu_k^K\|_2}{K}). \tag{11}$$

The degree of each prototype μ_j^K is $D_R(\mu_j^K) = \sum_{k=1}^{M} s(\mu_j^K, \mu_k^K)$. We sort the K-dimensional prototype representations in $\mathbf{PR}^K$ according to their degree $D_R(\mu_j^K)$, and rearrange $\widehat{X}_p^K$ and $\widehat{A}_p^K$ accordingly.

As we have stated in Section II-B, in this work we employ the K-dimensional DB representations [32] as the initialized vectorial vertex representations to compute the K-level correspondence matrix (C_p^K) of each graph G_p. Specifically, the DB representation can encapsulate rich nested structure information from each local vertex to the global graph structure through the K-layer expansion subgraphs rooted at the vertex. To construct reliable grid structures for graphs with rich multi-scale structure information, we vary the parameter K from 1 to L ($K \leq L$) and compute the final **aligned vertex grid structure** for each graph $G_p \in \mathbf{G}$ as

$$\widehat{X}_p = \sum_{K=1}^{L} \frac{\widehat{X}_p^K}{L}, \tag{12}$$

and the associated **aligned grid vertex adjacency matrix** as

$$\widehat{A}_p = \sum_{K=1}^{L} \frac{\widehat{A}_p^K}{L}, . \tag{13}$$

Remarks: Eq.(12) and Eq.(13) transform the original graphs $G_p \in \mathbf{G}$ with varying number of nodes $|V_p|$ into a new aligned grid graph structure with the same number of vertices, where $\widehat{X}_p$ is the corresponding aligned grid vertex feature matrix and $\widehat{A}_p$ is the corresponding aligned grid vertex adjacency matrix. Since for any graph $G_p \in \mathbf{G}$ the rows of $\widehat{X}_p$ are consistently indexed by the same prototype representations, the fixed-sized vertex grid structure $\widehat{X}_p$ can be directly employed as the input of a traditional convolutional neural network. In other words, one can apply a fixed sized classical convolutional filter to slide over the rows of $\widehat{X}_p$ and learn the feature for $G_p \in \mathbf{G}$. Finally, note that $\widehat{X}_p$ and $\widehat{A}_p$ accurately encapsulate the original feature and structural information of G_p, respectively.

B. The Quantum Spatial Graph Convolution Layer

In this subsection, we propose a new quantum spatial graph convolution layer to further extract the features of the vertices of each graph. This is defined by quantum information propagation between aligned grid vertices. To this end, we employ the average mixing matrix of the continuous-time quantum walk on the associated aligned grid vertex adjacency matrix. For the sample graph $G_p(V_p, E_p)$, we pass the aligned vertex grid structure $\widehat{X}_p \in \mathbb{R}^{M\times c}$ and the associated aligned grid vertex adjacency matrix $\widehat{A}_p \in \mathbb{R}^{M\times M}$ of G_p as the input of the quantum spatial graph convolution layer. The proposed spatial graph convolution layer takes the following form

$$Z = \mathrm{Relu}(Q\widehat{X}_p W), \tag{14}$$

where Relu is the rectified linear units function (i.e., a non-linear activation function), Q is the average mixing matrix of the continuous-time quantum walk on $\widehat{A}_p$ of G_p defined in Section II-A, $W \in \mathbb{R}^{c\times c'}$ is the matrix of trainable parameters of the proposed graph convolutional layer, and $Z \in \mathbb{R}^{M\times c'}$ is the output activation matrix.

The proposed quantum spatial graph convolution layer defined by Eq.(14) consists of three steps. In the first step the operation $\widehat{X}_pW$ is applied to transform the aligned grid vertex information matrix into a new aligned grid vertex information matrix. This in turn maps the c-dimensional features of each aligned grid vertex into new $c^{'}$-dimensional features, i.e., $\widehat{X}_pW$ maps the c feature channels to $c^{'}$ channels in the next layer. The weights W are shared among all aligned grid vertices. The second step computes QY, where $Y := \widehat{X}_pW$. This propagates the feature information of each aligned grid vertex to the remaining vertices as well as the vertex itself, in terms of the vertex visiting information of quantum walks. Specifically, we note that Q_{ij} encapsulates the average probability for a continuous-time quantum walk to visit the j-th aligned grid vertex starting from the i-th aligned grid vertex, and $(Q\widehat{X}_p^{'})_i = \sum_j Q_{ij}Y_j$. Here, i can be equal to j, i.e., Q includes the self-loop information for each vertex. Thus, the i-th row of the resulting matrix of $Q\widehat{X}_p^{'}$ is the feature summation of the i-th aligned grid vertex and the remaining aligned grid vertices associated with the average visiting probability of quantum walks from the i-th vertex to the remaining vertices as well as the i-th vertex itself. The final step applies the rectified linear unit function to $Q\widehat{X}_pW$ and outputs the graph convolution result.

The proposed quantum spatial graph convolution propagates the aligned grid vertex information in terms of the vertex visiting information associated with the continuous-time quantum walk between vertices. To further extract the multi-scale features of the aligned grid vertices, we stack multiple graph convolution layers defined by Eq.(14) as follows

$$Z_{t+1} = \mathrm{Relu}(QZ_tW_t), \tag{15}$$

where Z_0 is the input aligned vertex grid structure $\widehat{X}_p$, $Z_t \in \mathbb{R}^{M\times c_t}$ is the output of the t-th spatial graph convolution layer, and $W_t \in \mathbb{R}^{c_t\times c_{t+1}}$ is the trainable parameter matrix mapping c_t channels to c_{t+1} channels.

After each t-th graph convolutional layer, we add a layer to horizontally concatenate the output Z^t associated with the outputs of the previous 1 to $t-1$ spatial graph convolutional layers and the original input Z^0 as $Z_{0:t}$, i.e., $Z_{0:t} = [Z_0, Z_1, \ldots, Z^t]$ and $Z_{0:t} \in \mathbb{R}^{M\times\sum_{z=0}^{t} c_z}$. As a result, for the concatenated output $Z_{0:t}$, each of its row can be seen as the new multi-scale features for the corresponding grid vertex.

Remarks: Note that the proposed quantum spatial graph convolution only extracts new features for the grid vertex and does not change the orders of the vertices. As a result, both the output Z^t and the concatenated output $Z_{0:t}$ preserve the grid structure property of the original input $Z_0 = \widehat{X}_p$, and can be directly employed as the input of the traditional convolutional neural network. This provides an elegant way of bridging the gap between the proposed quantum spatial graph convolution layer and the traditional convolutional neural network, making an end-to-end deep learning architecture that integrates the graph representation and learning in both the quantum spatial graph convolution layer and the traditional convolution layer for graph classification problems.

C. The Remaining Convolutional Neural Network Layers

After the t-th proposed quantum spatial graph convolution layers, we get a concatenated vertex grid structure $Z_{0:t} \in \mathbb{R}^{M\times\sum_{z=0}^{t} c_z}$, where each row of $Z_{0:t}$ represents the multi-scale feature for a corresponding grid vertex. As we mentioned above, each grid structure $Z_{0:t}$ can be directly employed as the input to the traditional convolutional neural network (CNN). Specifically, the Classical One-dimensional CNN part of Fig.1 exhibits the architecture of the traditional CNN layers associated with each $Z_{0:t}$. Here, each concatenated vertex grid structure $Z_{0:t}$ is seen as a $M \times 1$ (in Fig.1 $M = 5$) vertex grid structure and each vertex is represented by a $\sum_{z=0}^{t} c_z$-dimensional feature, i.e., the channel of each grid vertex is $\sum_{z=0}^{t} c_z$. Then, we add a one-dimensional convolutional layer. The convolutional operation can be performed by sliding a fixed-sized filter of size $k\times 1$ (in Fig.1 $k = 3$) over the spatially neighboring vertices. After this, several MaxPooling layers and remaining one-dimensional convolutional layers can be added to learn the local patterns on the aligned grid vertex sequence. Finally, when we vary t from 0 to T (in Fig.1 $T = 2$), we will obtain $T+1$ extracted pattern representations. We concatenate the extracted patterns of each $Z_{0:t}$ and add a fully-connected layer followed by a Softmax layer.

D. Advantages of the Proposed QSGCNN Model

The proposed QSGCNN model is related to some existing state-of-the-art graph convolution network models and graph kernels. However, there are a number of significant theoretical differences between the QSGCNN model and these existing methods, explaining the effectiveness of the proposed model.

First, similar to the quantum spatial graph convolution of the proposed QSGCNN model, the associated graph convolution of the Deep Graph Convolutional Neural Network (DGCNN) [21] and the spectral graph convolution of the Fast Approximate Graph Convolutional Neural Network (FAGCNN) [34] also propagate the features between the graph vertices. Specifically, the graph convolutions of the DGCNN and FAGCNN models use the graph adjacency matrix or the normalized Laplacian matrix to determine how to pass the information among the vertices. In contrast, our quantum spatial graph convolution utilizes the average mixing matrix of the continuous-time quantum walk associated with the graph. As we mentioned in Section II-A, the quantum walk is not dominated by the low frequency values of the Laplacian spectrum and thus has a better ability to distinguish different graph structures. As a result, the proposed method can extract more discriminative vertex features.

Second, in order to maintain the scale of the vertex features after each graph convolution layer, the graph convolution of the DGCNN model [21] and the spectral graph convolution of the FAGCNN model [34] need to perform a multiplication by the inverse of the vertex degree matrix. For instance, the graph convolution layer of the DGCNN model associated with a graph having n vertices is

$$Z = \mathrm{f}(\widetilde{D}^{-1}\widetilde{A}XW), \tag{16}$$

where $\widetilde{A} = A + I$ is the adjacency matrix of the graph with added self-loops, $\widetilde{D}$ is the degree matrix of $\widetilde{A}$, $X^{n\times c}$

is the vertex feature matrix with each row representing the c-dimensional features of a vertex, $W^{c\times c^{'}}$ is the matrix of trainable parameters, f is a nonlinear activation function (e.g., the Relu function), and $Z^{n\times c^{'}}$ is the output. In a manner similar to the proposed quantum spatial graph convolution defined in Eq.(14), XW maps the c-dimensional features of each vertex into a set of new $c^{'}$-dimensional features. Moreover, $\widetilde{A}Y$ ($Y := \widehat{X}_pW$) propagates the feature information of each vertex to its neighboring vertices as well as the vertex itself. The i-th row $(\widetilde{A}Y)_i$ of the resulting matrix $\widetilde{A}Y$ represents the extracted features of the i-th vertex, and corresponds to the summation of Y_i itself and Y_j from the neighbor vertices of the i-th vertex. Multiplying by the inverse of $\widetilde{D}$ (i.e., $\widetilde{D}^{-1}$) can be seen as the process of normalizing and assigning equal weights between the i-th vertex and each of its neighbours. In other words, the graph convolution of the DGCNN model considers the mutual-influences between specified vertices for the convolution operation as the same. In contrast, the quantum spatial graph convolution of the proposed QSGCNN model defined in Eq.(14) assigns an average quantum walk visiting probability distribution to specified vertices with each vertex having a different visiting probability as the weight. Therefore, the extracted vertex feature is the weighted summation of the specified vertex features. As a result, the quantum spatial graph convolution of the proposed QSGCNN model not only maintains the feature scale, but also discriminates the mutual-influences between specified vertices in terms of the different visiting probabilities during the convolution operation.

Third, similar to the proposed QSGCNN model, both the PATCHY-SAN based Graph Convolution Neural Network (PSGCNN) model [20] and the DGCNN model [21] need to rearrange the vertex order of each graph structure and transform each graph into the fixed-sized vertex grid structure. Specifically, the PSGCNN model first forms the grid structures and then performs the standard classical CNN on the grid structures. The DGCNN model sorts the vertices through a SortPooling associated with the extracted vertex features from multiple spatial graph convolution layers. Unfortunately, both the PSGCNN model and the DGCNN model sort the vertices of each graph based on the local structural descriptor, ignoring consistent vertex correspondence information between different graphs. By contrast, the proposed QSGCNN model associates with a transitive vertex alignment procedure to transform each graph into an aligned fixed-sized vertex grid structure. As a result, only the proposed QSGCNN model can integrate the precise structural correspondence information over all graphs under investigations.

Fourth, when the PSGCNN model [20] and the DGCNN model [21] form fixed-sized vertex grid structures, some vertices with lower ranking will be discarded. Moreover, the Neural Graph Fingerprint Network (NGFN) [18] and the Diffusion Convolution Neural Network (DCNN) [19] tend to capture global-level graph features by summing up the extracted local-level vertex features through a SumPooling layer, since both the NGFN model and the DCNN model cannot directly form vertex grid structures. This leads to significant information loss for local-level vertex features. By contrast, the required aligned vertex grid structures and the associated grid vertex adjacency matrices for the proposed QSGCNN model can encapsulate both the original vertex features and the topological structure information of the original graphs, i.e., computing the local-level vertex grid structures will not discard any vertex of original graphs. As a result, the proposed QSGCNN reduces the shortcoming of information loss arising in the mentioned state-of-the-art graph convolutional neural network models.

Fifth, similar to the DGCNN model [21], the quantum spatial graph convolution of the proposed QSGCNN model is also related to the Weisfeiler-Lehman subtree kernel (WLSK) [9]. Specifically, the WLSK kernel employs the classical Weisfeiler-Lehman (WL) algorithm as a canonical labeling method to extract multi-scale vertex features corresponding to subtrees for graph classification. The key idea of the WL method is to concatenate a vertex label with the labels of its neighbor vertices, and then sort the concatenated label lexicographically to assign each vertex a new label. The procedure repeats until a maximum iteration h, and each vertex label at an iteration h corresponds to a subtree of height t rooted at the vertex. If the concatenated label of two vertices are the same, the subtree rooted at the two vertices are isomorphic, i.e., the two vertices are seen to share the same structural characteristics within the graph. The WLSK kernel uses this idea to measure the similarity between two graphs. It uses the WL method to update the vertex labels, and then counts the number of identical vertex labels (i.e. counting the number of the isomorphic subtrees) until the maximum of the iteration h in order to compare two graphs at multiple scales. To exhibit the relationship between the proposed quantum spatial graph convolution defined in Eq.(14) and the WLSK kernel, we decompose Eq.(14) in a row-wise manner, i.e.,

$$Z_i = \mathrm{Relu}(Q_{i,:}Y) = \mathrm{Relu}(Q_{ii}Y_i + \sum_j Q_{ij}Y_j), \qquad (17)$$

where $Y = \widehat{X}_pW$. For Eq.(17), Y_i can be seen as the continuous valued vectorial vertex label of the i-th vertex. Moreover, if $Q_{ij} > 0$, the quantum walk starting from the i-th vertex can visit the j-th vertex, and the visiting probability is Q_{ij}. In a manner similar to the WL methods, Eq.(17) aggregates the continuous label Y_i of the i-th vertex and the continuous labels Y_j of the vertices, that can be visited by the quantum walk starting from the i-th vertex, as a new signature vector $Q_{ii}Y_i + \sum_j Q_{ij}Y_j$ for the i-th vertex. The Relu function maps $Q_{ii}Y_i+\sum_j Q_{ij}Y_j$ to a new continuous vectorial label. As a result, the quantum spatial graph convolution of the proposed QSGCNN model can be seen as **a quantum version of the WL algorithm**, in terms of the quantum vertex information propagation formulated by the quantum walk. As we mentioned in Section II-A, the quantum walk can significantly reduce the effect of the tottering problem. On the other hand, the classical WL method also suffers from tottering problem [11]. As a result, the quantum spatial graph convolution can address the tottering problem arising in the classical WL method, and the graph convolution of the DGCNN model is similar to the clasical WL method. In other words, the quantum spatial graph convolution of the proposed QSGCNN model can learn better vertex features of graphs.

Finally, note that the proposed QSGCNN model for each graph is invariant with respect to the permutation of the vertices, indicating that the activations of a pair of isomorphic graphs will be the same. As we mentioned, the proposed QSGCNN model consists of three stages, i.e., a) the grid structure construction and input layer, b) the quantum spatial graph convolution layer, and c) the traditional CNN layer. For the first layer, the construction of grid structures relies on the vertex features and adjacency matrix, and is invariant to vertex permutations. As a result, the grid structures for a pair of isomorphic graphs are the same. For the second layer, the input grid structures of different graphs share the same parameter weights, thus the quantum spatial graph convolutions will produce the same extracted vertex features for a pair of isomorphic graphs associated with the same grid structures. Consequently, the subsequent classical CNN layer will correctly identify the isomorphic graphs. As a result, the proposed QSGCNN model can correctly identify pairs of isomorphic graphs.

IV. EXPERIMENTS

In this section, we empirically compare the performance of the proposed QSGCNN model to state-of-the-art approaches. Specifically, we utilize nine benchmark graph datasets from bioinformatics [35], [36], [37], [38] and social networks [39] to evaluate the graph classification performance of the proposed QSGCNN model. These benchmark datasets include MUTAG, PTC, NCI1, PROTEINS, D&D, COLLAB, IMDB-B, IMDB-M and RED-B, and are all available on the website https://chrsmrrs.github.io/datasets. A selection of statistics of these datasets are shown in Table I. Note that, all the benchmark social network datasets (i.e., the COLLAB, IMDB-B, IMDB-M and RED-B datasets) used in this work consist of multiple graphs, and have been widely employed to evaluate the classification performance of existing graph convolutional neural network models and graph kernels.

A. Comparisons with Graph Kernels

Experimental Setup: We evaluate the performance of the proposed QSGCNN model on graph classification problems against eight alternative state-of-the-art graph kernels. These graph kernels include 1) Jensen-Tsallis q-difference kernel (JTQK) with $q = 2$ [25], 2) the Weisfeiler-Lehman subtree kernel (WLSK) [9], 3) the Weisfeiler-Lehman kernel based on core variants (CORE WL) [40], 4) the shortest path graph kernel (SPGK) [41], 5) the shortest path kernel based on core variants (CORE SP) [40], 6) the random walk graph kernel (RWGK) [42], 7) the graphlet count kernel (GK) [43], and 8) the propagated information graph kernel (PIGK) [44].

For the evaluation, **the proposed QSGCNN model uses the same network structure on all graph datasets**. We commence by setting the number of prototype representations to $M = 64$, since we observe that about 60% to 70% of the graphs have less than 64 vertices in our experiments. This can guarantee that the proposed QSGCNN model not only preserves all original vertices, but also retains the independent edge connections between vertices as much as possible. In other words, most edge connections between vertices will not be merged into one edge during the process of transforming each arbitrary sized graph into the fixed-sized grid structure. Moreover, we set the number of the quantum spatial graph convolution layers as 5 (note that, including the original input grid structures, the spatial graph convolution produces 6 concatenated outputs), and the channels of each quantum spatial graph convolution as 32. Following each of the concatenated outputs after the quantum graph convolution layers, we add a traditional CNN layer with the architecture as C64-P2-C64-P2-C64-F64 to learn the extracted patterns, where Ck denotes a traditional convolutional layer with k channels, Pk denotes a classical MaxPooling layer of size and stride k, and Fk denotes a fully-connected layer consisting of k hidden units. The filter size and stride of each Ck are all 5 and 1. With the six sets of extracted patterns after the CNN layers to hand, we concatenate them and add a new fully-connected layer followed by a Softmax layer with a dropout rate of 0.5. We use the rectified linear units (ReLU) in either the graph convolution or the traditional convolution layer. The learning rate of the QSGCNN model is 0.0005 for all datasets. The only hyperparameter we optimized is the number of epochs and the batch size for the mini-batch gradient descent algorithm. To optimize the QSGCNN model, we use the Stochastic Gradient Descent with the Adam updating rules. Finally, note that, the QSGCNN model needs to construct the prototype representations to identify the transitive vertex alignment information over all graphs. The prototype representations can be computed from the training graphs or both the training and testing graphs. We observe that the QSGCNN model associated with the two variants does not influence the final performance. Thus, in our evaluation we proposed to compute the prototype representations from both the training and testing graphs. In this sense, our model can be seen as an instance of transductive learning [45], where all the graphs are used to compute the prototype representations, and the class labels of the test graphs are not observed during the training phase. For the QSGCNN model, we perform 10-fold cross-validation to compute the classification accuracies, with nine folds for training and one folds for testing. For each dataset, we repeat the experiment 10 times and report the average classification accuracies and standard errors in Table II.

We set the parameters controlling the maximum height of the subtrees for the Weisfeiler-Lehman isomorphism test (WL-SK kernel) and for the tree-index method (JTQK kernel) to 10. This is based on the previous empirical studies of Shervashidze et al. [9] and Bai et al. [25]. For each graph kernel, we perform 10-fold cross-validation using the LIBSVM implementation of C-Support Vector Machines (C-SVM) and we compute the classification accuracies. We perform cross-validation on the training data to select the optimal parameters for each kernel and fold. We repeat the experiment 10 times for each kernel and dataset and report the average classification accuracies in Table II. Note that for some kernels we directly report the best results from the original corresponding papers, since the evaluation of these kernels followed the same setting of ours. Note that, the symbol $-$ in Table II indicates that some approaches were not evaluated on the corresponding datasets

TABLE I
INFORMATION OF THE GRAPH DATASETS

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC(MR)	COLLAB	IMDB-B	IMDB-M	RED-B
Max # vertices	28	111	620	5748	109	492	136	89	3783
Mean # vertices	17.93	29.87	39.06	284.30	25.60	74.49	19.77	13.00	429.61
Mean # edges	19.79	32.30	72.82	715.65	14.69	4914.99	193.06	131.87	497.80
# graphs	188	4110	1113	1178	344	5000	1000	1500	2000
# vertex labels	7	37	61	82	19	–	–	–	–
# classes	2	2	2	2	2	3	2	3	2
Description	Chemical	Chemical	Chemical	Chemical	Chemical	Social	Social	Social	Social

by the original authors, and this symbol has the same meaning in the following Table III and Table IV

Experimental Results and Discussion: Table II shows that the proposed QSGCNN model significantly outperforms the alternative state-of-the-art graph kernels in this study. Although, the proposed model cannot achieve the best classification accuracy on the NCI1 and COLLAB datasets, the proposed model is still competitive and the accuracy on the COLLAB dataset is only a little lower than the WL-OA kernel. On the other hand, the accuracy of the proposed model on the NCI1 dataset is still higher than the SPGK, CORE SP, GK and RWGK kernels. The reasons for the effectiveness are twofold. First, the state-of-the-art graph kernels for comparisons are typical examples of R-convolution kernels. Specifically, these kernels are based on the isomorphism measure between any pair of substructures, ignoring the structure correspondence information between the substructures. By contrast, the associated aligned vertex grid structure for the proposed QSGCNN model incorporates the transitive alignment information between vertices over all graphs. Thus, the proposed model can better reflect the precise characteristics of graphs. Second, the C-SVM classifier associated with graph kernels can only be seen as a shallow learning framework [46]. By contrast, the proposed QSGCNN model can provide an end-to-end deep learning architecture for graph classification, and can better learn the graph characteristics. The experiments demonstrate the advantages of the proposed QSGCNN model, compared to the shallow learning framework. Third, some alternative kernels are related to the Weisfeiler-Lehman method. As we have stated in Section III-D, the kernels based on the Weisfeiler-Lehman method may suffer from the tottering problem. By contrast, the proposed model based on quantum walk can significantly reduce the effect of tottering walks. The experiments also demonstrate the effectiveness.

B. Comparisons with Deep Learning Methods

Experimental Setup: We evaluate the performance of the proposed QSGCNN model on graph classification problems against eleven alternative state-of-the-art deep learning methods for graphs. These methods include 1) the deep graph convolutional neural network (DGCNN) [21], 2) the PATCHY-SAN based convolutional neural network for graphs (PSGCNN) [20], 3) the diffusion convolutional neural network (DCNN) [19], 4) the edge-conditioned convolutional networks (ECC) [47], 5) the deep graphlet kernel (DGK) [48], 6) the graph capsule convolutional neural network (GCCNN) [49], 7) the anonymous walk embeddings based on feature driven (AWE) [50], 8) the graph convolution network based on Differentiable Pooling (DiffPool) [51], 9) the graph convolution network based on Self-Attention Pooling (SAGPool) [52], 10) the graph convolutional network with EigenPooling (EigenPool) [52], and 11) the degree-specific graph neural networks (DEMO-Net) [53]. For the proposed QSGCNN model, we use the same experimental setups when we compare the proposed model to graph kernels. For the PSGCNN, ECC, and DGK model, we report the best results from the original papers [20], [47], [48]. Note that, these methods follow the same setting with the proposed QSGCNN model. For the DCNN model, we report the best results from the work of Zhang et al. [21], following the same setting of our network. For the AWE model, we report the classification accuracies of the feature-driven AWE, since the authors have stated that this kind of AWE model can achieve competitive performance on label dataset. Moreover, the PSCN and ECC models can leverage additional edge features. Since most graph datasets and all the alternative methods used for comparisons do not leverage edge features, in this work we do not report the results associated with edge features. Finally, since the SAGPool, EigenPool, DEMO-Net models have not been evaluated on the social network datasets by the original authors, and ECC and the DiffPool models are only evaluated on one social network dataset (i.e., the COLLAB dataset) by the original author where the accuracies (67.79 and 75.48) are obviously lower than ours. For fair comparisons, we only report the accuracies of these models on the bioinformatics datasets in Table.IV.

TABLE IV
CLASSIFICATION ACCURACY FOR COMPARISONS WITH DEEP LEARNING METHODS ON BIOINFORMATICS DATASETS.

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC
QSGCNN	**91.32**	**77.50**	75.90	**81.70**	**63.37**
DiffPool	82.66	76.00	76.25	80.64	–
SAGPool	–	74.06	71.86	76.45	–
EigenPool	79.50	77.00	**78.60**	76.60	–
DEMO-Net	81.40	–	–	70.80	57.20

Finally, note that, in order to further demonstrate the advantage of the proposed QSGCNN model associated with quantum walks, we perform the proposed spatial convolutional operation associated with classical random walks. More specifically, for the proposed QSGCNN model and its associated spatial graph convolutional operation function $Z = \mathrm{Relu}(Q\widehat{X}_pW)$ defined by Eq.14, we replace the quantum average mixing matrix Q by $\widehat{D}_p^{-1}\widehat{A}_p$, where $\widehat{A}_p$ is the aligned vertex adjacency matrix, $\widehat{D}_p^{-1}$ is inverse of the degree matrix for $\widehat{A}_p$, $P = \widehat{D}_p^{-1}\widehat{A}_p$ is the transition matrix of the classical random walk, and $P(i,j)$ represents the probability of a random walk starting from vertex v_i to vertex v_j. As a result, the revised convolutional operation $Z = \mathrm{Relu}(\widehat{D}_p^{-1}\widehat{A}_p\widehat{X}_pW)$ will propagate the feature information between aligned grid vertices based on the vertex visiting information of classical random walks. We report the results of the neural network model based on classical random walks (CSGCNN) following the same network architecture and experimental setup as for the proposed QSGCNN model.

All the classification accuracies and standard errors for each deep learning method are shown in Table.III.

Experimental Results and Discussion: Table III indicates

TABLE II
CLASSIFICATION ACCURACY (IN % ± STANDARD ERROR) FOR COMPARISONS WITH GRAPH KERNELS.

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC(MR)	COLLAB	IMDB-B	IMDB-M	RED-B
QSGCNN	**91.32**±0.91	77.50±0.91	**75.90**±0.79	**81.70**±0.92	**63.37**±1.15	**78.80**±0.89	73.62±1.12	**51.60**±1.15	**91.50**±0.24
JTQK	85.50±0.55	**85.32**±0.14	72.86±0.41	79.89±0.32	58.50±0.39	76.85±0.40	72.45±0.81	50.33±0.49	77.60±0.35
WLSK	82.88±0.57	84.77±0.13	73.52±0.43	79.78±0.36	58.26±0.47	77.39±0.35	71.88±0.77	49.50±0.49	76.56±0.30
CORE WL	87.47±1.08	85.01±0.19	–	79.24±0.34	59.43±1.20	–	**74.02**±0.42	51.35±0.48	78.02±0.23
SPGK	83.38±0.81	74.21±0.30	75.10±0.50	78.45±0.26	55.52±0.46	58.80±0.2	71.26±1.04	51.33±0.57	84.20±0.70
CORE SP	88.29±1.55	73.46±0.32	–	77.30±0.80	59.06±0.93	–	72.62±0.59	49.43±0.42	90.84±0.14
PIGK	76.00±2.69	82.54±0.47	73.68±0.69	78.25±0.51	59.50±2.44	–	–	–	–
GK	81.66±2.11	62.28±0.29	71.67±0.55	78.45±0.26	52.26±1.41	72.83±0.28	65.87±0.98	45.42±0.87	77.34±0.18
RWGK	80.77±0.72	63.34±0.27	74.20±0.40	71.70±0.47	55.91±0.37	–	67.94±0.77	46.72±0.30	–

TABLE III
CLASSIFICATION ACCURACY (IN % ± STANDARD ERROR) FOR COMPARISONS WITH GRAPH CONVOLUTIONAL NEURAL NETWORKS.

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC(MR)	COLLAB	IMDB-B	IMDB-M	RED-B
QSGCNN	**91.32**±0.91	77.50±0.91	75.90±0.79	**81.70**±0.92	63.37±1.15	**78.80**±0.89	**73.62**±1.12	**51.60**±1.15	**91.50**±0.24
DGCNN	85.83±1.66	74.44±0.47	75.54±0.94	79.37±0.94	58.59±2.47	73.76±0.49	70.03±0.86	47.83±0.85	76.02±1.73
PSGCNN	88.95±4.37	76.34±1.68	75.00±2.51	76.27±2.64	62.29±5.68	72.60±2.15	71.00±2.29	45.23±2.84	86.30±1.58
DCNN	66.98	56.61±1.04	61.29±1.60	58.09±0.53	56.60	52.11±0.71	49.06±1.37	33.49±1.42	–
ECC	76.11	76.82	72.65	74.10	–	67.79	–	–	–
GCCNN	–	**82.72**±2.38	76.40±4.71	77.62±4.99	**66.01**±5.91	77.71±2.51	71.69±3.40	48.50±4.10	87.61±2.51
DGK	82.66±1.45	62.48±0.25	71.68±0.50	78.50±0.22	57.32±1.13	73.09±0.25	66.96±0.56	44.55±0.52	78.30±0.30
AWE	87.87±9.76	–	–	71.51±4.02	–	70.99±1.49	73.13±3.28	51.58±4.66	82.97±2.86
CSGCNN	88.65±0.76	75.51±0.25	73.24±0.50	78.68±0.47	62.58±0.90	77.96±0.90	72.50±0.55	50.00±0.95	89.15±0.24

that the proposed QSGCNN model significantly outperforms state-of-the-art deep learning methods for graph classifications, on the MUTAG, D&D, COLLAB, IBDM-B, IBDM-M and RET-B datasets. On the other hand, only the accuracy of the GCCNN model on the NCI1 and PTC datasets and that of the DGCNN model on the PROTEINS dataset are a higher than the proposed QSGCNN model. But the proposed QSGCNN is still competitive and outperform the remaining methods on the three datasets. The reasons of the effectiveness are fivefold.

First, similar to the state-of-the-art graph kernels, all the alternative deep learning methods (i.e., the DGCNN, PSGCNN, DCNN, GCCNN, DGK, AWE, HO-GCN, ECC, SAGPool, EigenPool and DEMO-Net models) for comparisons also cannot integrate the correspondence information between graphs into the learning architecture. Especially, the PSGCNN, DGCNN and ECC models need to reorder the vertices, but these methods rely on simple but inaccurate heuristics to align the vertices of the graphs, i.e., they sort the vertex orders based on the local structure descriptor of each individual graph and ignore the vertex correspondence information between different graphs. Thus, only the QSDCNN model can reflect the graph characteristics through the layer-wise learning.

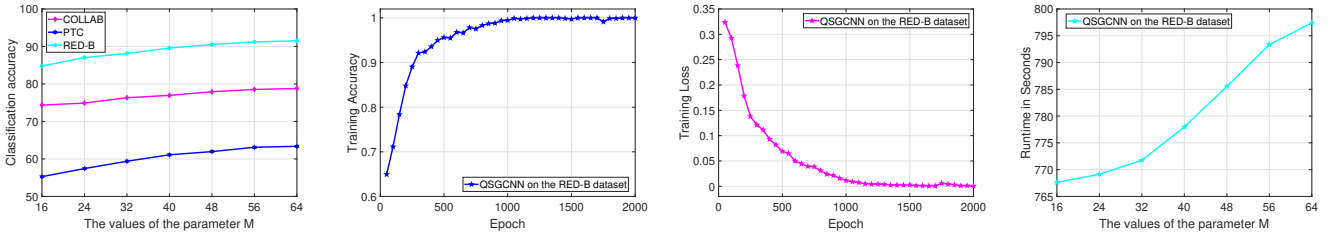
Second, the PSGCNN and DGCNN models need to form a fixed-sized vertex grid structure for each graph. Since the vertex numbers of different graphs are different, forming such fixed-sized grid structures means some vertices of each graph may be discarded, leading to information loss. By contrast, as we have mentioned in Section II and Section III, the associated aligned vertex grid structures can completely preserve the information of original graphs. As a result, only the proposed QSGCNN model can completely integrate the original graph characteristics into the learning process.

Third, the DCNN model needs to sum up the extracted local-level vertex features from the convolution operation as global-level graph features through a SumPooling layer. By contrast, the QSGCNN model can learn the graph topological information through the local vertex features.

Forth, unlike the DGCNN, PSGCNN, DCNN, GCCNN, DGK, AWE, HO-GCN, ECC, SAGPool, EigenPool and DEMO-Net models that are based on the original vertex adjacency matrix to formulate vertex connection information of the graph convolution operation, the graph convolution operation of the proposed QSGCNN model formulates the vertex connection information in terms of the average mixing matrix of continuous-time quantum walk. As we have stated in Section II, the quantum walk is not dominated by the low frequency of the Laplacian spectrum and can better distinguish different graph structures. Thus, the proposed model has better ability to identify the difference between different graphs.

Fifth, similar to the DGCNN, PSGCNN, DCNN, HO-GCN, DEMO-Net and DGK models, the proposed QSGCNN model is also related to the classical Weisfeiler-Lehman (WL) method. Since the classical WL method suffers from tottering problem, the related DGCNN, PSGCNN and DGK models also possess the same drawback. By contrast, the graph convolution operation of the proposed QSGCNN model can be seen as the quantum version of the classical WL algorithm. Since the quantum walk can reduce the tottering problem, the proposed QSGCNN model overcomes the shortcoming of tottering problem arising in the DGCNN, PSGCNN and DGK models. Moreover, the AWE model is based on the classical random walk. By contrast, the proposed QSGCNN model is based on the quantum random walk, that has been proven powerful to better distinguish different graph structures. The evaluation demonstrates the advantages of the QSGCNN model, compared to the state-of-the-art deep learning methods.

Finally, we observe that the proposed QSGCNN model significantly outperforms the CSGCNN model, that is based on the proposed spatial graph convolutional operation associated with classical random walks. This indicates that the proposed QSGCNN model associated with quantum walks can better discriminate different graph structures than the CSGCNN model associated with classical random walks, demonstrating the advantage of utilizing quantum walks in our framework. On the other hand, excluding the QSGCNN model, we observe that the CSGCNN model can outperform most of the alternative methods. This is because the CSGCNN model follows the same architecture and experimental setup with the QSGCNN model, i.e., the CSGCNN model also employs the fixed-sized grid structures of graphs as inputs. Similar to the proposed QSGCNN model, the CSGCNN model can also reduce the problem of information loss and overcome the shortcoming of lacking structure correspondence information, demonstrating the advantage of the proposed fixed-sized grid structures.



(a) Test accuracy vs parameter M (b) Training accuracy vs epoch (c) Training loss vs epoch (d) Training runtime vs parameter M

Fig. 3. Plots of accuracy and runtime vs parameter M, as well as accuracy and training loss vs epoch.

C. Computational Efficiency of the Proposed Model

In this subsection, we empirically evaluate the computational efficiency of QSGCNN, and compare it with the fast WLSK kernel [9] on the RED-B benchmark dataset. We choose this dataset because the graphs it contains have the largest average size among the available datasets used in our experimental evaluation. The WLSK kernel takes $2,170$ seconds to compute the kernel matrix, and another 837 seconds to train the C-SVM associated with the kernel matrix for one round of 10-fold cross validation. For the proposed QSGCNN model, computing the fixed-sized grid structures takes $3,980$ seconds, and another 457 seconds to train the QSGCNN model for one round of 10-fold cross validation. Note that, the training time of the proposed QSGCNN model depends on the number of epochs selected. Here we set the number of epochs to be 100. QSGCNN can significantly outperform the WLSK kernel under in this setting, i.e., the accuracies of QSGCNN and the WLSK kernel are 77.30 versus 76.56. As a result, the overall runtimes for QSGCNN and the WLSK kernel are 4437 seconds versus $3,007$ seconds. In other words, although the runtime of the proposed QSGCNN model is slightly higher, this is still a competitive advantage when compared with that of the WLSK kernel. More importantly, the proposed QSGCNN model can significantly outperform the WLSK kernel in terms of graph classification accuracy, i.e., QSGCNN provides a better trade-off between classification accuracy and computational efficiency.

D. Additional Performance Evaluation of the Proposed Model

In this subsection, we first evaluate how the selection of the parameter M influences the classification performance of the proposed model. Specifically, we vary the parameter M from 16 to 64 (with steps of size 8). Figure 3(a) shows how the classification accuracy of the proposed QSGCNN model varies with increasing values of M on the COLLAB, PTC and RED-B datasets. We select these three datasets due to their representativeness in terms of different graph size and graph sample size. We observe that the classification accuracy gradually increases as the value of M increases, reaching a more stable value when M is greater than 48. Moreover, we evaluate how the training loss and the classification accuracies vary as we increase the number of epochs. Specifically, we vary the epoch number from 50 to 1000 (with steps of size 50). We show the results in Figure 3(b) and Figure 3(c). Finally, we investigate how the runtime varies with the above increasing values of M when set the epoch as 1000. We show the results in Figure 3(d). Note that although the above three evaluations are only performed on the RED-B dataset, we observe similar results on the remaining datasets. As the number of epochs increases, the classification accuracy gradually increases and the training loss gradually decreases, until they both converge to stable values. Moreover, the runtime increases approximately linearly with increasing values of M.

V. Conclusion

In this paper we have developed a new quantum graph convolutional neural network, QSGCNN, that can directly learn an end-to-end deep learning architecture for classifying graphs of arbitrary sizes. The key idea is to present a novel quantum spatial graph convolution operation on a fixed-sized vertex grid structure for the original graphs. This transformation is achieved through transitive alignments between graphs. We demonstrate that the proposed QSGCNN model not only preserves the original graph characteristics, but also bridges the gap between the spatial graph convolution layer and the traditional convolutional neural network layer. Moreover, QSGCNN can better distinguish different structures, and the experiments demonstrate its effectiveness on graph classification problems.

In previous work [54], [55] we have shown how to characterize edge information in the original graphs through directed line graphs, where each vertex of the line graph represents an edge of the original graph. We have also illustrated the relationship between discrete-time quantum walks and the directed line graphs. It would be interesting to develop a novel quantum edge-based convolutional network associated with discrete-time quantum walks using the directed line graph. Finally, Xu et al. [56] indicate that the convolutional operation underpinning most graph convolutional networks based on an adjacency matrix representation can be interpreted as directly implementing a 1-layer perceptron followed by a non-linear activation function. Moreover, they develop a new graph isomorphism network model based on a vertex information aggregation layer followed by multi-layer perceptrons, and demonstrate a significant performance improvement. This can inform our future work, and we will further extend QSGCNN to develop a new quantum isomorphism dectection network.

References

[1] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of NIPS*, 2016, pp. 3837–3845.

[2] W. Wu, B. Li, L. Chen, X. Zhu, and C. Zhang, "K-ary tree hashing for fast graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 936–949, 2018.

[3] L. Bai and E. R. Hancock, "Graph kernels from the Jensen-Shannon divergence," *Journal of Mathematical Imaging and Vision*, vol. 47, no. 1-2, pp. 60–69, 2013.

[4] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of KDD*, 2016, pp. 1225–1234.

[5] K. Riesen and H. Bunke, "Graph classification by means of lipschitz embedding," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 39, no. 6, pp. 1472–1483, 2009.

[6] R. C. Wilson, E. R. Hancock, and B. Luo, "Pattern vectors from algebraic graph theory," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 7, pp. 1112–1124, 2005.

[7] M. Neuhaus and H. Bunke, *Bridging the Gap between Graph Edit Distance and Kernel Machines*, ser. Series in Machine Perception and Artificial Intelligence. WorldScientific, 2007, vol. 68.

[8] D. Haussler, "Convolution kernels on discrete structures," in *Technical Report UCS-CRL-99-10*, Santa Cruz, CA, USA, 1999.

[9] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-Lehman graph kernels," *Journal of Machine Learning Research*, vol. 1, pp. 1–48, 2010.

[10] N. Kriege and P. Mutzel, "Subgraph matching kernels for attributed graphs," in *Proceedings of ICML*, 2012.

[11] L. Bai, L. Rossi, Z. Zhang, and E. R. Hancock, "An aligned subtree kernel for weighted graphs," in *Proceedings of ICML*.

[12] L. Bai, Z. Zhang, C. Wang, X. Bai, and E. R. Hancock, "A graph kernel based on the Jensen-Shannon representation alignment," in *Proceedings of IJCAI*, 2015, pp. 3322–3328.

[13] L. Lu, Y. Zheng, G. Carneiro, and L. Yang, Eds., *Deep Learning and Convolutional Neural Networks for Medical Image Computing - Precision Medicine, High Performance and Large-Scale Datasets*, ser. Advances in Computer Vision and Pattern Recognition. Springer, 2017.

[14] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of CVPR*, 2015, pp. 3156–3164.

[15] A. J. Tixier, G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Classifying graphs as images with convolutional neural networks," *CoRR*, vol. abs/1708.02218, 2017. [Online]. Available: http://arxiv.org/abs/1708.02218

[16] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Proceddings of NIPS*, 2015, pp. 2449–2457.

[17] J. Vialatte, V. Gripon, and G. Mercier, "Generalizing the convolution operator to extend CNNs to irregular domains," *CoRR*, vol. abs/1606.01166, 2016. [Online]. Available: http://arxiv.org/abs/1606.01166

[18] D. K. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proceedings of NIPS*, 2015, pp. 2224–2232.

[19] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proceedings of NIPS*, 2016, pp. 1993–2001.

[20] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of ICML*, 2016, pp. 2014–2023.

[21] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of AAAI*, 2018.

[22] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *CoRR*, vol. abs/1812.04202, 2018. [Online]. Available: http://arxiv.org/abs/1812.04202

[23] E. Farhi and S. Gutmann, "Quantum computation and decision trees," *Physical Review A*, vol. 58, p. 915, 1998.

[24] M. M. Wilde, *Quantum Information Theory*. Cambridge University Press, 2017.

[25] L. Bai, L. Rossi, H. Bunke, and E. R. Hancock, "Attributed graph kernels using the Jensen-Tsallis q-differences," in *Proceedings of ECML-PKDD*, 2014, pp. 99–114.

[26] K. M. Borgwardt, "Graph kernels," Ph.D. dissertation, Ludwig Maximilians University Munich, Germany, 2007.

[27] X. Shuai, Y. Ding, J. R. Busemeyer, Y. Sun, S. Chen, and J. Tang, "Does quantum interference exist in twitter?" *CoRR*, vol. abs/1107.0681, 2011.

[28] M. Melucci, "Relevance feedback algorithms inspired by quantum detection," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1022–1034, 2016.

[29] ——, "An efficient algorithm to compute a quantum probability space," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1452–1463, 2019.

[30] A. Fawaz, P. Klein, S. Piat, S. Severini, and P. Mountney, "Training and meta-training binary neural networks with quantum computing," in *Proceedings of KDD*, 2019, pp. 1674–1681.

[31] C. Godsil, "Average mixing of continuous quantum walks," *Journal of Combinatorial Theory, Series A*, vol. 120, no. 7, pp. 1649–1662, 2013.

[32] L. Bai, L. Cui, L. Rossi, L. Xu, and E. Hancock, "Local-global nested graph kernels using nested complexity traces," *Pattern Recognit. Lett.*, vol. 134, pp. 87–95, 2020.

[33] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2011.

[34] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[35] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "Brenda, the enzyme database: updates and major new developments," *Nucleic Acids Research*, vol. 32, no. Database-Issue, pp. 431–433, 2004.

[36] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of Medicinal Chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[37] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. Mol. Biol.*, vol. 330, no. 4, p. 771C783, 2003.

[38] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowl. Inf. Syst.*, vol. 14, no. 3, pp. 347–375, 2008.

[39] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2008. [Online]. Available: http://graphkernels.cs.tu-dortmund

[40] G. Nikolentzos, P. Meladianos, S. Limnios, and M. Vazirgiannis, "A degeneracy framework for graph similarity," in *Proceedings of IJCAI*, 2018, pp. 2595–2601.

[41] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Proceedings of the IEEE International Conference on Data Mining*, 2005, pp. 74–81.

[42] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *Proceedings of ICML*, 2003, pp. 321–328.

[43] N. Shervashidze, S. Vishwanathan, K. M. T. Petri, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," *Journal of Machine Learning Research*, vol. 5, pp. 488–495, 2009.

[44] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, "Propagation kernels: efficient graph kernels from propagated information," *Machine Learning*, vol. 102, no. 2, pp. 209–245, 2016.

[45] A. Gammerman, K. S. Azoury, and V. Vapnik, "Learning by transduction," in *Proceedings of UAI*, 1998, pp. 148–155.

[46] S. Zhang, C. Liu, K. Yao, and Y. Gong, "Deep neural support vector machines for speech recognition," in *Proceedings of ICASSP*, 2015, pp. 4275–4279.

[47] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of CVPR*, 2017, pp. 29–38.

[48] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proceedings of KDD*, 2015, pp. 1365–1374.

[49] S. Verma and Z. Zhang, "Graph capsule convolutional neural networks," *CoRR*, vol. abs/1805.08090, 2018. [Online]. Available: http://arxiv.org/abs/1805.08090

[50] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," in *Proceedings of ICML*, 2018, pp. 2191–2200.

[51] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Processing of NeurIPS*, 2018, pp. 4805–4815.

[52] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proceedings of ICML*, 2019, pp. 3734–3743.

[53] J. Wu, J. He, and J. Xu, "Demo-net: Degree-specific graph neural networks for node and graph classification," in *Proceedings of KDD*, 2019.

[54] L. Bai, L. Rossi, L. Cui, Z. Zhang, P. Ren, X. Bai, and E. R. Hancock, "Quantum kernels for unattributed graphs using discrete-time quantum walks," *Pattern Recognition Letters*, vol. 87, pp. 96–103, 2017.

[55] L. Bai, F. Escolano, and E. R. Hancock, "Depth-based hypergraph complexity traces from directed line graphs," *Pattern Recognition*, vol. 54, pp. 229–240, 2016.

[56] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *CoRR*, vol. abs/1810.00826, 2018. [Online]. Available: http://arxiv.org/abs/1810.00826

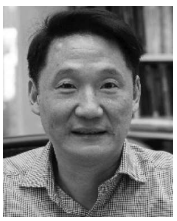
Lu Bai Lu Bai received the Ph.D. degree from the University of York, UK, and both the B.Sc. and M.Sc degrees from Macau University of Science and Technology, Macau SAR, China. He was a recipient of the National Award for Outstanding Self-Financed Chinese Students Study Aboard by China Scholarship Council in 2015, and the Best Paper Awards of the International Conferences ICIAP 2015 (Eduardo Caianello Best Student Paper Award) and ICPR 2018. He is now an Associate Professor in Central University of Finance and Economics, Beijing, China. He has published more than 80 journal and conference papers, including TPAMI, TNNLS, TCYB, PR, ICML, IJCAI, ECML-PKDD, ICDM, etc. His current research interests include pattern recognition, machine learning, and financial data analysis. He is currently a member of the editorial board of the journal Pattern Recognition.

Yuhang Jiao Yuhang Jiao received both the B.Sc and M.Sc from Central University of Finance and Economics (CUFE), Beijing, China. He was supervised by Dr. Lu Bai and Dr. Lixin Cui for pursuing his M.Sc. He has published more than 10 journal and conference papers, including TPAMI, IJCAI, ECML-PKDD, ICPR, GbRPR and S+SSPR. His current research interests include graph-based structured data analysis.

Lixin Cui Lixin Cui received the Ph.D. degree from the University of Hong Kong, HKSAR, China, and both the B.Sc. and M.Sc. degrees from Tianjin University, Tianjin, China. She is now an Associate Professor in Central University of Finance and Economics, Beijing, China. She was the recipient of the Outstanding Paper Awards of the International Conference IEEE IEEM 2019, the Best Student Paper Awards of the International Conferences APIEMS 2011 and WCE 2011. She is currently an Associate Editor of Pattern Recognition Journal. She has published more than 40 journal and conference papers, including TPAMI, TFS, TCYB, TNNLS, PR, WWWJ, IJCAI, ECML-PKDD, etc. Her current research interests include machine learning, deep learning, and their applications in Fintech problems. She is currently a member of the editorial board of the journal Pattern Recognition.

Luca Rossi Luca Rossi received the PhD degree in computer science from the Ca Foscari University of Venice, Italy, in 2013. He is currently a lecturer with the Queen Mary University of London, UK, having held various positions with the University of Birmingham, UK, Aston University, UK, and the Southern University of Science and Technology, China. He has published more than 50 papers in international journals and conferences. His research interests include the areas of pattern recognition, data mining, and network science. He is currently a member of the editorial board of the journal Pattern Recognition.

Yue Wang Yue Wang received the Ph.D. degree from Sichuan University, Sichuan, China, and both the B.Sc. and M.Sc. degrees from Hefei University of Technology, Anhui, China. He was a postdoctor of Peking University, Beijing, China. He is now an Associate Professor in School of Information, Central University of Finance and Economics, Beijing, China. His current research interests include data mining and machine learning.



Philip S. Yu Philip S. Yu received received the B.S. degree in electrical engineering from National Taiwan University, the M.S. and Ph.D. degrees in EE from Stanford University, and the MBA degree from New York University. He is currently a Distinguished Professor of computer science with the University of Illinois at Chicago (UIC), and holds the Wexler Chair in information technology. He has published more than 970 papers in refereed journals and conferences. He holds or has applied for over 300 US patents. He was a member of the Steering Committee of the IEEE Data Engineering and the IEEE Conference on Data Mining. He is a Fellow of the ACM and the IEEE. He is on the Steering Committee of the ACM Conference on Information and Knowledge Management. He received the ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of big data, the IEEE Computer Society's 2013 Technical Achievement Award for "pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining, and anonymization of big data", and the Research Contributions Award from ICDM 2003, for his pioneering contributions to the field of data mining. He also received the ICDM 2013 10-year Highest-Impact Paper Award, and the EDBT Test of Time Award (2014). He has received several IBM honors, including two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 94th plateau of Invention Achievement Awards. He was the Editor-in-Chief of the IEEE Transactions on Knowledge and Data Engineering (2001-2004).

Edwin R. Hancock Edwin R. Hancock (F16) received the B.Sc., Ph.D., and D.Sc. degrees from the University of Durham, Durham, UK. He is currently an Emeritus Professor with the Department of Computer Science, University of York, York, UK. He has published over 200 journal articles and 650 conference papers. Prof. Hancock was a recipient of the Royal Author Biography Society Wolfson Research Merit Award in 2009, the Pattern Recognition Society Medal in 1991, the BMVA Distinguished Fellowship in 2016 and the IAPR Piere Devijver Award in 2018. He is a fellow of the IAPR, IEEE, the Royal Astronomical Society, the Institute of Physics, the Institute of Engineering and Technology, and the British Computer Society. He was named Distinguished Fellow by the British Machine Vision Association. He has also received best paper prizes at CAIP 2001, ACCV 2002, ICPR in 2006 and 2018, BMVC 20 07, ICIAP in 2009 and 2015. He is currently Editor-in-Chief of the journal Pattern Recognition, and was founding Editor-in-Chief of IET Computer Vision from 2006 until 2012. He has also been a member of the editorial boards of the journals IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Recognition, Computer Vision and Image Understanding, Image and Vision Computing, and the International Journal of Complex Networks. He has been Conference Chair for BMVC in 1994 and Program Chair in 2016, Track Chair for ICPR in 2004 and 2016 and Area Chair at ECCV 2006 and CVPR in 2008 and 2014, and in 1997 established the EMMCVPR workshop series. He was Second Vice President of the International Association of Pattern Recognition (2016-2018). He is currently an IEEE Computer Society Distinguished Visitor (2021-2023).