Learning Aligned Vertex Convolutional Networks for Graph Classification

Lixin Cui, Lu Bai, Xiao Bai, Yue Wang, Edwin R. Hancock, *IEEE Fellow*

***Abstract*—Graph Convolution Networks (GCNs) are powerful tools for graph structure data analysis. One main drawback arising in most existing GCN models is that of the over-smoothing problem, i.e., the vertex features abstracted from existing graph convolution operation have previously tended to be indistinguishable if the GCN model has many convolutional layers (e.g., more than 2 layers). To address this problem, in this paper we propose a family of Aligned Vertex Convolutional Network (AVCN) models that focuses on learning multi-scale features from local-level vertices for graph classification. This is done by adopting a transitive vertex alignment algorithm to transform arbitrary sized graphs into fixed-sized grid structures. Furthermore, we define a new aligned vertex convolution operation that can effectively learn multi-scale vertex characteristics by gradually aggregating local-level neighboring aligned vertices residing on the original grid structures into a new packed aligned vertex. With the new vertex convolution operation to hand, we propose two architectures for the AVCN models to extract different hierarchical multi-scale vertex feature representations for graph classification. We show that the proposed models can avoid iteratively propagating redundant information between specific neighboring vertices, restricting the notorious over-smoothing problem arising in most spatial-based Graph Convolution Network (GCN) models. Experimental evaluations on benchmark datasets demonstrate the effectiveness.**

***Index Terms*—Graph Neural Networks, Graph Convolution Networks, Vertex Convolution, Graph Classification**

I. Introduction

There have been increasing interests to generalize Convolutional Neural Networks (CNNs) [1], [2], [3], [4], [5] to graph domains. These neural networks on graphs are now widely known as Graph Convolutional Networks (GCNs) [6], [7], [8], [9], [10], and have been proven effective to extract highly meaningful statistical features from graph structures [11]. The aim of this work is to develop novel GCN models to learn rich features of local-level vertices for graph classification.

A. Literature Review

Broadly speaking, most existing GCN models are developed based on one of two strategies, i.e., either a) spectral or b) spatial strategy. Approaches based on the former strategy employ the Fourier domain properties of the convolution operator based on graph spectral theory [12], [13], [14], [15]. However, since most approaches based on the spectral strategy request the graph sizes to be identical, these approaches can not accommodate arbitrary sized graphs and thus different Fourier bases. Hence, the spectral-based approaches are usually utilized for vertex classification. Approaches based on the latter strategy, on the other hand, extend the standard convolution operation of classical CNN models into graph structures by propagating feature information between spatially adjacent vertices [16], [17], [18]. Since spatial-based approaches do not demand graph structures to be the same sizes, these approaches can be directly adopted for graph classification. However, the performance of most spatial-based approaches are relatively low on graph classification. The reason for this ineffectiveness is that these approaches tend to directly compute the summation of the feature information of local-level vertices from the convolution operation as global characteristics of graph structures through a SumPooling layer. Hence, the local topological information residing on the vertices may be substantially discarded.

To address this drawback of the GCN models associated with SumPooling, a number of spatial-based GCN models focusing on local-level vertex information have been proposed. For example, Niepert et al. [19] have proposed a novel GCN model by re-ordering the vertices and converting each graph into a fixed-sized vertex grid structure, where standard one-dimensional CNNs can be directly used. Zhang et al. [20] have developed a Subgraph Convolutional Network model by re-ordering the vertices and constructing the fixed-sized expansion subgraph based grid structure rooted at each vertex, so that a standard one-dimensional convolution can be directly slid over the subgraph based grid structures to extract multi-scale features for the corresponding rooted vertices. Similarly, Zhang et al. [21] have proposed a Deep Graph Convolutional Neural Network (DGCNN) model, that reserves rich local information of vertices by using global characteristics of graph topologies. To this end, a SortPooling layer is proposed to construct fixed-sized vertex grid structures through the unordered vertex features abstracted from convolution layers. Then traditional convolution operations can be directly performed on the grid structures to further extract the multi-scale feature information. These aforementioned approaches focus more on local-level vertex features and achieve better performance than most existing GCN models for graph classification. However, they tend to sort the vertex order based on each individual graph. As a result, they can not

Lixin Cui (cuilixin@cufe.edu.cn), Lu Bai (Corresponding Author: bailucs@cufe.edu.cn), and Yue Wang (wangyuecs@cufe.edu.cn) are with Engineering Research Center of State Financial Security, Ministry of Education, Central University of Finance and Economics, Beijing, 102206, China. Xiao Bai is with School of Computer Science and Engineering, Beihang University, Beijing, China (baixiao@buaa.edu.cn). Edwin R. Hancock (edwin.hancock@york.ac.uk) is with 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), the program for innovation research in Central University of Finance and Economics, and the Emerging Interdisciplinary Project of CUFE.

easily reflect accurate structural correspondence information between graphs. Furthermore, these methods also suffer from the problem of information loss. This usually arises when they operate with a fixed-sized vertex grid structure and thus low rank vertices are discarded.

To address the drawbacks of the aforementioned spatial-based GCN models, Bai et al. [22], [23] have proposed a family of Aligned-Spatial Graph Convolutional Network models (i.e., the ASGCN model [22] and its backtrackless version BASGCN model [23]) to learn local-level vertex features through aligned fixed-sized grid structures. The grid structures encapsulate the transitive correspondence information between graphs, and are constructed using the original vertex feature and vertex adjacency matrices without discarding vertices. Thus, unlike existing spatial-based GCN models, both ASGCN and BASGCN models can either reduce the information loss problem or better reflect the structural correspondence information. Unfortunately, all the graph convolution operations of the DGCNN, ASGCN and BASGCN models rely on information propagation between neighboring vertices indicated by the adjacency matrix. As a result, these models may suffer from the over-smoothing problem [6], i.e., the vertex features abstracted from the graph convolution operation tend to be more and more similar and are indistinguishable after multiple convolutional layers [24]. In other word, these GCN models may fail to capture diversified local vertex information, influencing their performance on graph or vertex classification problems. Generally speaking, developing effective GCN approaches for graphs is still a challenging problem.

B. Contributions of This Work

The main objective of this paper is to overcome the drawbacks of the aforementioned spatial-based GCN methods by proposing novel Aligned Vertex Convolutional Network (AVCN) models for graph classification. To inherit the effectiveness of the aforementioned GCN models, the starting point is to convert arbitrary sized graphs into fixed-sized aligned vertex-based grid structures through the associated transitive vertex alignment procedure of the aforementioned ASGCN and BASGCN models [22], [23], replacing the original vertex feature and adjacency matrices of each graph. This not only guarantees the consistency between the spatial positions and structural correspondences of vertices over all graphs, but also provides a way of developing a novel aligned vertex convolution operation to abstract multi-scale aligned vertex features from the grid structures. Overall, the main novel contributions of this paper are threefold.

First, with the aforementioned fixed-sized aligned vertex-based grid structure of each graph to hand, we develop a new aligned vertex convolution operation. This is done by adopting a fixed-sized one-dimensional convolution filter on the grid structure which slides across the entire set of ordered aligned vertices. We show that the proposed vertex convolution operation can effectively learn multi-scale vertex characteristics by gradually aggregating together similar local-level aligned vertices residing on the original grid structure as a new aligned vertex, and thus produce a new packed grid structure with a reduced number of packed aligned vertices. By contrast, the convolution operation of most existing spatial-based GCN models (e.g., the ASGCN as well as the DGCNN models) extract new characteristics for each vertex by repeatedly propagating redundant vertex feature information between its neighboring vertices indicated by the adjacency matrix, and thus remain the original vertex numbers. As a result, the resulting AVCN models associated with the new vertex convolution operation can avoid iteratively propagating redundant information between specific neighboring vertices and significantly restrict the aforementioned over-smoothing problem. Moreover, since the packed aligned vertex after each convolution operation is extracted by aggregating the aligned vertices residing on the original grid structures with specific spatial positions. The resulting AVCN models can maintain the consistency between the spatial positions and structural correspondences for the extracted grid structures after the convolution operation, reflecting precise structural correspondence information between graphs during the convolution operation.

Second, to extract different hierarchical multi-scale feature representations of the aligned vertices, we propose two architectures for the AVCN models associated with the new vertex convolution operation. Both architectures are defined associated with a family of parallel stacked vertex convolution layers consisting with multiple vertex convolution filters of different sizes (see Figure 4 for more details). As a result, the proposed AVCN models can reflect rich hierarchical multi-scale local-level vertex features of each graph structure.

Third, we empirically investigate the performance of the proposed models on graph classification problems. Experiments on benchmark datasets demonstrate the effectiveness.

This paper is organized as follows. Section II reviews related works. Section III presents how to convert arbitrary sized graphs into fix-sized aligned vertex grid structures. Section IV defines the new AVCN models. Section V provides experimental evaluations. Section VI gives conclusions.

II. Review of Related Spatial-based GCN Models

We commence by briefly reviewing two representative spatial-based GCN models described elsewhere in the literature. To this end, we introduce the associated graph convolution operations for the Deep Graph Convolutional Neural Network (DGCNN) model [21] as well as the Aligned-Spatial Graph Convolutional Network (ASGCN) model [22]. To commence, we assume a sample graph $G(V,E)$ drawn from a graph set $\mathbf{G}$, where V is the set of vertices, E is the set of edges, $X=(x_1,x_2,...,x_n)\in\mathbb{R}^{n\times c}$ encapsulates the c-dimensional (i.e., $n=|V|$) feature vectors of the n vertices from G, and $A\in\mathbb{R}^{n\times n}$ is the vertex adjacency matrix. Note that, A can be either a weighted or an un-weighted adjacency matrix. If G is a vertex attributed graph, X can be the one-hot encoding matrix associated with the vertex labels. If G is an un-attributed graph, we adopt the vertex degrees as the corresponding labels of vertices.

A. The DGCNN Model

With the sample graph $G(V,E)$ to hand, the spatial graph convolution operation of the DGCNN model is defined as

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

where $\tilde{A} = A + I$ represents the adjacency matrix of G associated with pre-added self-loop connections, $\tilde{D}$ refers to the degree matrix of $\tilde{A}$ ($\tilde{A}_{i,i} = \sum_j \tilde{A}_{i,j}$), $W \in \mathbb{R}^{c\times c^{'}}$ represents the matrix of trainable parameters for graph convolution, f is a nonlinear activation function, and $Z \in \mathbb{R}^{n\times c^{'}}$ is the output.

The above graph convolution procedure is mainly comprised of four computational steps. The first step computes the matrix product XW which converts the c-dimensional feature vector of each vertex into a new $c^{'}$-dimensional feature vector, where all vertices share the same filter weights W. The second step computes the matrix product $\tilde{A}Y$ (where $Y := XW$) which propagates the feature information between adjacent vertices. Here, the i-th row $(\tilde{A}Y)_{i,:}$ 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. The third step multiplies the matrix $\tilde{A}Y$ by the inverse of $\tilde{D}$ (i.e., $\tilde{D}^{-1}$) to normalize each row of $\tilde{A}Y$. This step can be seen as the process of keeping a fixed feature scale after the vertex feature propagation (i.e., the graph convolution operation), by assigning equal weights $\tilde{D}_{i,i}$ between the i-th vertex and its neighbouring vertices. The final step applies a nonlinear activation function and outputs the convolution result.

Remarks: Eq.(1) shows that the associated spatial graph convolution procedure of the DGCNN model fails to distinguish the importance of different vertices when it performs the convolution operation. The reason for this is that the feature transformations of different vertices rely on the same filter weight matrix W. Thus, the trainable weight matrix W of the DGCNN model can not directly affect the feature aggregation process. Actually, this drawback also appears in other spatial-based GCN models, such as the Neural Graph Fingerprint Network (NGFN) model [17] the Diffusion Convolution Neural Network (DCNN) model [18], and the Quantum Spatial Graph Convolutional Neural Network (QSGCNN) model [25]. The convolution procedures of these GCN models also follow a similar form with that of the DGCNN model. The trainable parameters of the underlying convolution operations are shared by each individual vertex. Moreover, these spatial-based GCN models suffer from over-smoothing, i.e., the vertex features abstracted from the graph convolution operation tend to be indistinguishable or similar if the GCN model has more than 2 convolutional layers [6], [24]. Since the required graph convolution operation of these GCN models relies on the feature information propagation between neighboring vertices indicated by the vertex adjacency matrix $\tilde{A}$. This process may propagate redundant information between adjacent vertices, taking place multiple times over the multiple convolutional layers. Clearly, these two drawbacks limit the effectiveness of state-of-the-art spatial-based GCN models. □

B. The ASGCN Model

To avoid ignoring the influence of different vertices in the aforementioned spatial-based GCN models, we have developed the ASGCN model that can adaptively discriminate vertex importance [22]. For the graph $G(V,E)$, we commence by converting its vertex feature matrix X and its associated vertex adjacency matrix $\tilde{A}$ into the fixed-sized aligned vertex grid structure $\bar{X} \in \mathbb{R}^{M\times c}$ (i.e., the aligned grid vertex feature matrix) and the associated aligned grid vertex adjacency matrix $\bar{A} \in \mathbb{R}^{M\times M}$. Here the vertices of the same spatial position are also transitively matched (see details in Section III). The spatial graph convolution operation of the ASGCN model is defined as the following form

$$Z^h = \mathrm{f}(\bar{D}^{-1}\bar{A}\sum_{j=1}^{c}(\bar{X}\odot W^h)_{:,j}). \tag{2}$$

Here, f is a nonlinear activation function, $W^h \in \mathbb{R}^{M\times c}$ is the trainable graph convolution parameter matrix of the h-th convolution filter of filter size $M\times 1$ and channel number c, $\odot$ is the element-wise Hadamard product, $\bar{D}$ is the degree matrix of $\bar{A}$, and $Z \in \mathbb{R}^{M\times 1}$ is the output matrix.

We observe that the mathematical form of the graph convolution operation for the ASGCN model in Eq.(2) is similar to that for the DGCNN model defined in Eq.(1), and also consists of four main corresponding computational steps. The only difference between the two convolution operations is their first step. Specifically, for the DGCNN model, the first step of its convolution operation takes the form $Y := XW$ and maps the c-dimensional feature vector of each vertex into a new $c^{'}$-dimensional feature vector through the same filter weight matrix $W \in \mathbb{R}^{c\times c^{'}}$. In contrast, the ASGCN model takes the form $\bar{Y} := \sum_{j=1}^{c}(\bar{X}\odot W^h)_{:,j}$ as the first step of its convolution opeartion. The procedure $\sum_{j=1}^{c}(\bar{X}\odot W^h)_{:,j}$ first calculates the element-wise Hadamard product between $\bar{X}$ and W^h, and then compute the sum of the columns of $\bar{X}\odot W^h$) as $\bar{Y}$. The resulting matrix $\bar{Y}$ can be considered as a new weighted aligned vertex grid structure with one vertex feature channel, and the i-th aligned grid vertex residing on the i-th row of $\bar{X}$ is assigned by a different weighted vector $w_{i,:}$. Unlike the DGCNN model, the trainable parameter matrix W^h of the ASGCN model has a direct impact on the process of the vertex feature aggregation.

Remarks: Although the graph convolution procedure of the ASGCN model indicated by Eq.(2) addresses the shortcoming of overtaking the importance of different vertices in most existing spatial-based GCN models, it also suffers from over-smoothing. Similar to the DGCNN model, the convolution operation of ASGCN also relies on vertex feature information aggregations through its associated aligned grid vertex adjacency matrix $\bar{A}$, and this influences its effectiveness. On the other hand, for the backtrackless version of the ASGCN model (i.e., the BASGCN model [23]), the convolution operation takes the same as Eq.(2). The only difference between the two models is that the BASGCN model further transforms the original undirected adjacency matrix $\bar{A}$ into the directed adjacency matrix to restrict the tottering problem arising in the ASGCN model (see the previous work [23] for more details).

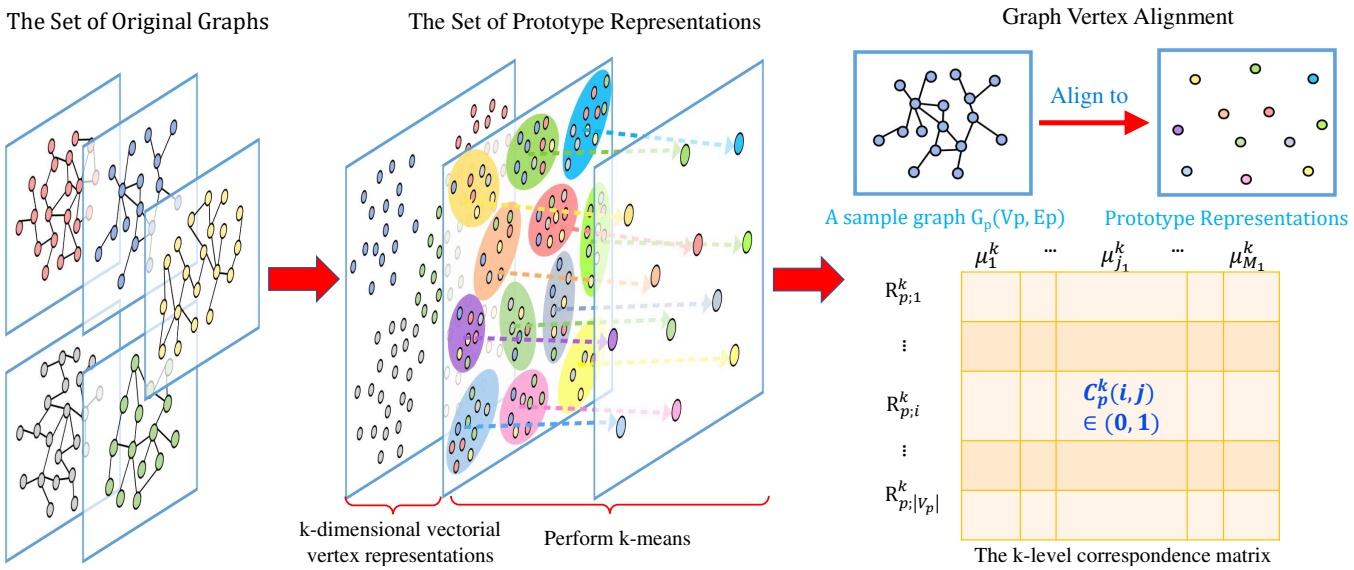


Fig. 1. Based on the previous work in [22], the procedure of transitively aligning vertices and computing the vertex correspondence matrix consists of three sequential computational stages. (1) We commence by representing the n vertices of all graphs in $\mathbf{G}$ as the k-dimensional vectorial representations $\mathbf{R}^k = \{\mathrm{R}_1^k, \mathrm{R}_2^k, \ldots, \mathrm{R}_n^k\}$. (2) We adopt the classical lite k-means clustering method [26] to divide the k-dimensional vectorial representations $\mathbf{R}^k$ of all graphs into M clusters. We compute the centroid point vectors of the M clusters and employ them as a set of prototype representations $\mathbf{PR}^k = \{\mu_1^k, \cdots, \mu_j^k, \cdots, \mu_M^k\}$, where each μ_j^k is the centroid point of the j-cluster. (3) We align the k-dimensional vectorial vertex representations of each graph $G_p \in \mathbf{G}$ to the prototype representations $\mathbf{PR}^k$ and compute a k-level correspondence matrix C_p^k to record the correspondence information between G_p and $\mathbf{PR}^k$. Let $v_i \in V_p$ be the i-th vertex of graph G_p. If the j-th prototype representation μ_j^k of $\mathbf{PR}^k$ is the nearest one to the k-dimensional vectorial representation $\mathrm{R}_{p;i}^k$ of vertex $v_i \in V_p$ regarding the Euclid distance in the vectorial principle space (i.e., $\mathrm{R}_{p;i}^k$ belongs to the j-th cluster identified in the second step), we say that the i-th vertex v_i of G_p is aligned to μ_j^k. In this case, we set $C_p^k(i,j) = 1$ to indicate the structure correspondence.

As a result, the performance of the BASGCN model may also be influenced by over-smoothing. In this paper, we propose a new variant of the Aligned Vertex Convolution Network model to overcome the above problems. □

III. Construct Aligned Vertex Grid Structures

To inherit the effectiveness of existing GCN models, we propose to define novel AVCN models based on the associated aligned vertex grid structures of the aforementioned ASGCN model [22]. In this section, we briefly introduce how to convert arbitrary sized graphs into the grid structures.

A. The Transitive Vertex Alignment Method

We commence by reviewing the transitive vertex alignment method developed in the previous work [22]. The main idea of this method is based on aligning the vertices of each graph to a family of prototype representations. Since the prototype representations are identified by locating the M centroids over the vectorial vertex representations of all graphs under evaluation using the classical k-means clustering method. The prototype representations can encapsulate main characteristics of the set of graphs under study. Specifically, assume the set of graphs is $\mathbf{G} = \{G_p(V_p, E_p), p = 1, ..., N\}$, where p is the graph index, V_p is the vertex set of the sample graph G_p, and E_p is the edge set. Fig. 1 exhibits the detailed alignment procedure for computing the **vertex correspondence matrix** based on the transitive vertex matching method.

Like our previous work in [22], we utilize the k-dimensional depth-based (DB) representations as the original k-dimensional vectorial vertex representations $\mathbf{R}^k$ to calculate the family of k-dimensional prototypes. Although, one can adopt any other method to initialize the vectorial representations of vertices [27], [15]. The DB representation is computed by gauging the entropy on the layered expansion subgraph rooted at each vertex [28]. Thus, the DB representation can contain significant entropy-based content flow rooted from each local vertex to the global structure of each original graph. Fig.2 shows the details of calculating the DB representation.

B. The Aligned Vertex-based Grid Structure

We now illustrate how to convert arbitrary-sized graphs into fixed-sized aligned vertex-based grid structures, where the vertices at the same corresponding spatial position are also transitively matched to each other. For the set of graphs $\mathbf{G}$ defined earlier, assume $X_p \in \mathbb{R}^{|V_p| \times c}$ and $\tilde{A}_p \in \mathbb{R}^{|V_p| \times n}$ are the original vertex feature matrix and the original vertex adjacency matrix (with the added self-loops) of a sample graph $G_p(V_p, E_p) \in \mathbf{G}$, respectively. With the k-level correspondence matrix C_p^k of G_p based on the definition in Section III-A to hand, we calculate the k-level aligned vertex feature matrix for G_p as

$$\bar{X}_p^k = (C_p^k)^T X_p, \tag{3}$$

where $\bar{X}_p^k \in \mathbb{R}^{M \times c}$, the rows of $\bar{X}_p^K$ are indexed by the corresponding prototypes in $\mathbf{PR}^k$, and each row of $\bar{X}_p^k$ represents the vectorial feature of an aligned vertex. We compute the k-level aligned vertex adjacency matrix for G_p as

$$\bar{A}_p^k = (C_p^k)^T (\tilde{A}_p)(C_p^k), \tag{4}$$

where $\bar{A}_p^k \in \mathbb{R}^{M \times M}$. Both the rows and the columns of $\bar{A}_p^k$ are indexed by the same prototypes in $\mathbf{PR}^k$. The matrix $\bar{A}_p^k$ **can also be viewed as the k-level aligned vertex feature**

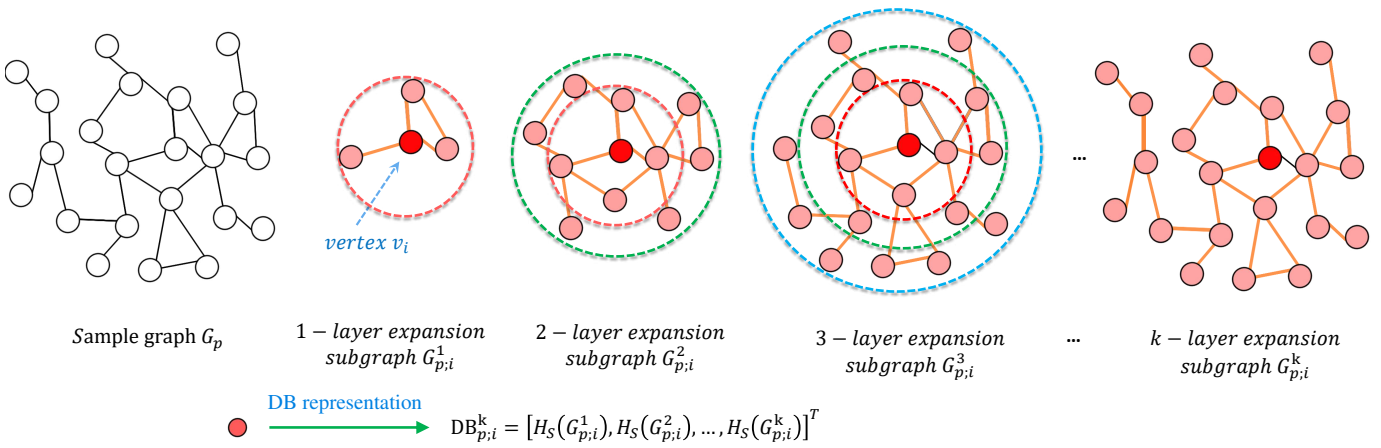


Fig. 2. The procedure to calculate the DB representation of a vertex [28]. Specifically, assume a sample graph $G_p(V_p, E_p) \in \mathbf{G}$ marked by black color, v_i is its i-th vertex (marked by red color). For v_i, we first compute its 1-th order neighborhood set $\mathcal{N}_i^1$ as $\mathcal{N}_i^1 = \{v_j \in V_p \mid d_s(v_i, v_j) \leq 1\}$, where $d_s(v_i, v_j)$ represents the shortest path length between j-th vertex v_j and i-th vertex v_i. The resulting 1-layer expansion subgraph $\mathcal{G}^1_{p;i}$ rooted at v_i is defined as the substructures (surrounded by the red broken line) associated with the vertices in $\mathcal{N}_i^1$ and the edges between the vertices from the original graph G_p. Analogously, we also abstract the 2-layer subgraph $\mathcal{G}^2_{p;i}$ (surrounded by the green line) and the 3-layer expansion subgraph $\mathcal{G}^3_{p;i}$ (surrounded by the blue broken line), respectively. Consequently, we establish a family of $\widetilde{k}$-layer expansion subgraphs rooted at v_i ($\widetilde{k} \in [1, k]$). Clearly, if k is greater than the length of the longest shortest path rooted from v_i to the remaining vertices, the k-layer expansion subgraph $\mathcal{G}^k_{p;i}$ is the global graph G_p itself. As a result, the k-dimensional DB representation of v_i is defined as $\mathrm{DB}^k_{p;i} = \{H_S(\mathcal{G}^1_{p;i}), \cdots, H_S(\mathcal{G}^{\widetilde{k}}_{p;i}), \cdots, H_S(\mathcal{G}^k_{p;i})\}^T$, where $H_S(\cdot)$ is the classical Shannon entropy of a subgraph based on classical random walks [29].

matrix of G_p, where each row of $\bar{A}_p^k$ contains the adjacency information of a corresponding aligned vertex to the remaining aligned vertices.

To construct the fixed-sized grid structure of each graph $G_p \in \mathbf{G}$ based on $\bar{X}_p^k$ and $\bar{A}_p^k$, we need to determine the spatial position of corresponding aligned vertices that are all indexed by the same set of prototype representations $\mathbf{PR}^k$. To this end, Bai et al., [22] have proposed to compute the Gaussian kernel-based similarity [30] between the prototype representations in $\mathbf{PR}^k$, and sort the prototypes based on the summation of the similarities between each prototype and the remaining ones. Then we can permute the elements of $\bar{X}_p^k$ and $\bar{A}_p^k$ accordingly, i.e., we determine the spatial positions of the aligned vertices based on the orders of their indexed prototypes. The above process is equivalent to sorting the prototypes in order of average similarity to the remaining ones. As a result, the aligned vertices indexed by the similar prototypes will be assigned to the spatial positions with close spatial proximity.

As stated in Section III-A, we use the k-dimensional DB representations as the vectorial vertex representation to construct the family of prototype representations $\mathbf{PR}^k$. Here, the DB representation is computed by measuring the entropies on a family of $\widetilde{k}$-layer ($\widetilde{k} \leq k$) expansion subgraphs rooted at each vertex [28]. When we vary the largest layer k of the expansion subgraphs from 1 to K (i.e., $k \leq K$), we can compute two kinds of aligned vertex-based grid structures associated with the k-level aligned vertex feature matrix $\bar{X}_p^k \in \mathbb{R}^{M \times c}$ and the k-level aligned vertex adjacency matrix $\bar{A}_p^k \in \mathbb{R}^{M \times M}$, respectively. Specifically, for each graph G_p, we compute the final **aligned vertex feature-based grid structure** as

$$\bar{X}_p^{\mathcal{F}} = \sum_{k=1}^{K} \frac{\bar{X}_p^k}{L}, \tag{5}$$

and the final **aligned vertex adjacency information-based grid structure** as

$$\bar{X}_p^{\mathcal{A}} = (\bar{D}_p)^{-1} \bar{A}_p, \tag{6}$$

where $\bar{A}_p = \sum_{k=1}^{K} \frac{\bar{A}_p^k}{L}$ is the mixed aligned vertex adjacency matrix, and $\bar{D}_p$ is its degree matrix, $\bar{X}_p^{\mathcal{F}} \in \mathbb{R}^{M \times c}$, $\bar{X}_p^{\mathcal{A}} \in \mathbb{R}^{M \times M}$, and the i-th rows of $\bar{X}_p^{\mathcal{F}}$ and $\bar{X}_p^{\mathcal{A}}$ correspond to the feature vector of the i-th aligned grid vertex. The aligned vertex-based grid structure $\bar{X}_p^{\mathcal{F}} \in \mathbb{R}^{M \times c}$ preserves the original vertex feature matrix. The aligned vertex-based grid structure $\bar{X}_p^{\mathcal{A}} \in \mathbb{R}^{M \times M}$ encapsulates the original adjacency information between each vertex to the remaining vertices as well as the vertex transition information arising in the DGCNN and ASGCN model. Moreover, $(\bar{D}_p)^{-1} \bar{A}_p$ indicates how the vertex features propagate between neighboring vertices during the convolution process.

Remarks: Since both the aligned vertex-based grid structures $\bar{X}_p^{\mathcal{F}}$ and $\bar{X}_p^{\mathcal{A}}$ are computed by converting the original feature and adjacency information of each vertex $v_p \in V_p$ to that of the new aligned vertices, they preserve the original vertex feature and structural information of G_p. This reduces the aforementioned information loss problem of existing graph convolutional network models [19], [21]. Moreover, since both $\bar{X}_p^{\mathcal{F}}$ and $\bar{X}_p^{\mathcal{A}}$ are computed by employing the transitive alignment procedure, they are indexed by the prototype representations from $\mathbf{PR}^k$ with consistent orders. Thus, we can guarantee that the aligned grid vertices at the same spatial position are also transitively matched to each other.

IV. The Aligned Vertex Convolutional Networks

In this section, we define the new Aligned Vertex Convolutional Network (AVCN) models for graph classification. We employ the transitive alignment information over a family of graphs and convert arbitrary sized graphs into fixed-sized vertex aligned grid structures. We then define an aligned vertex convolution operation by using a set of fixed-sized

TABLE I
IMPORTANT TERMS AND NOTATIONS

Symbol	Definitions
node e	the e-th vertex
$Z_{e,h}^{t}$	the h-th feature channel of vertex (e) in layer t
$W^{t,h,s}$	the filter that maps to the h-th feature channel in layer t from the s-th feature channel in layer t-1
$W_j^{t,h,s}$	the j-th element of the filter that maps to the h-th feature channel in layer t from the s-th feature channel in layer t-1
$b^{t,h}$	the bias of the h-th filter in layer t
σ	the activate function, e.g., Relu function
c_{t-1}	the number of filters in layer t-1

one-dimensional convolution filters on the aligned grid structure. With the new vertex convolution operation to hand, the proposed model can extract the original aligned vertex grid structure as a new grid structure with a reduced number of packed aligned vertices. As a result, the extracted multi-scale vertex features learned through the convolutional operation is packed into the new grid structure. Finally, we employ the Softmax layer to read-out the abstracted vertex features and predict the categories of graph structures.

A. The Aligned Vertex Convolution Operation

In this subsection, we develop a new AVCN model that learns local-level vertex features for graph classification. This model is defined by employing a set of fixed-sized one-dimensional convolution filters on the predefined aligned vertex-based grid structures and sliding the filter over the ordered aligned vertices to learn features, in a manner analogous to the standard convolution operation. Specifically, for each graph $G(V,E) \in \mathbf{G}$ defined earlier, we first compute its associated aligned vertex-based grid structure $\bar{X}$, based on the definition in Section III-B. Note that, $\bar{X}$ can be either the aligned vertex feature-based grid structure $\bar{X}_p^{\mathcal{F}} \in \mathbb{R}^{M\times c}$ (i.e., M aligned vertices each with c feature channels) or the aligned vertex adjacency information-based grid structure $\bar{X}_p^{\mathcal{A}} \in \mathbb{R}^{M\times M}$ (i.e., M aligned vertices each with M feature channels). We denote the element of $\bar{X}$ in the e-th row and s-th column as $\bar{X}_{e,s}$, i.e., the s-th feature channel of the e-th aligned vertex. We pass $\bar{X}$ to the convolution layer. Assume the size of the receptive field is m, i.e., the size of the one-dimensional convolution filter is m, the vertex convolution operation associated with 1-stride takes the form

$$Z_{e,h} = \sigma\Big(\sum_{s=1}^{c}\Big(\sum_{j=1}^{m} W_j^{h,s}\bar{X}_{e+j-1,s}\Big) + b^h\Big), \tag{7}$$

where $Z_{e,h}$ is the element in the e-th row and h-th column of the new grid structure Z after the convolution operation. The row index e satisfies the condition $e \le M-m+1$. The j-th convolution filter element $W_j^{h,s}$ maps the s-th feature channel of X to the h-th feature channel of Z, b^h is the bias of the h-th convolution filter, and σ is the activation function.

An example of the vertex convolution operation defined by Eq.(7) is shown in Fig. 3. The vertex convolution operation consists of two computational steps. In the first step, the convolution filter $\sum_{s=1}^{c}(\sum_{j=1}^{m} W_j^{h,s}\bar{X}_{e+j-1,s})$ is applied to map the e-th aligned vertex $\bar{X}_{e,:}$ as well as its neighbor vertices $\bar{X}_{e+j-1,:}$ $(j = 2,3)$ into a new feature value, associated

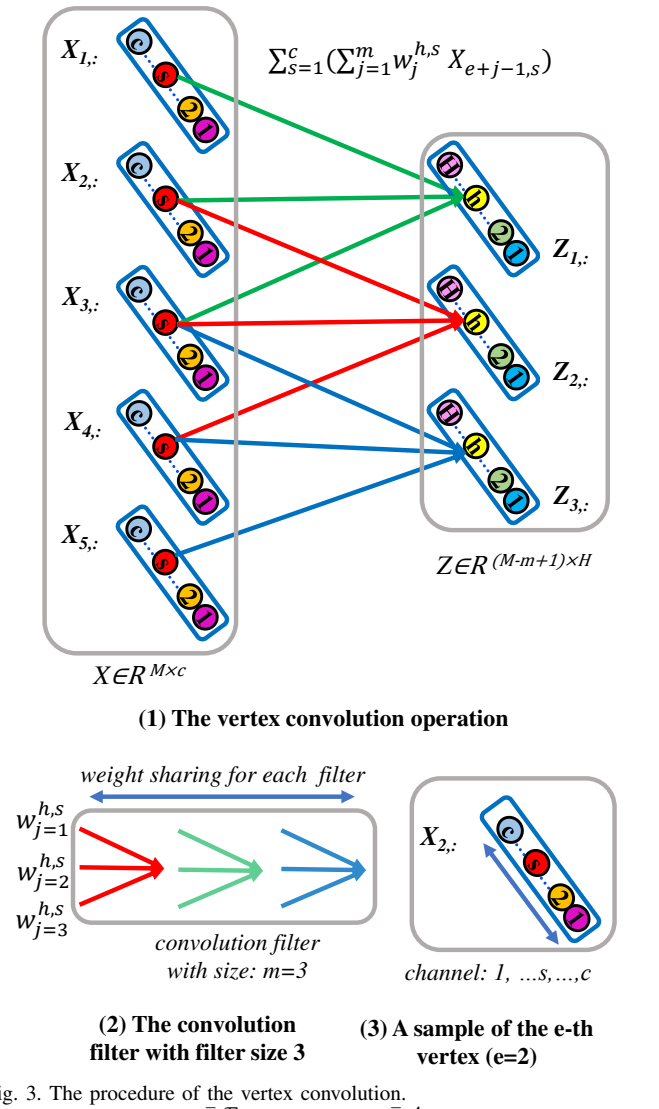


Fig. 3. The procedure of the vertex convolution.

with all the c (for $\bar{X}_p^{\mathcal{F}}$) or M (for $\bar{X}_p^{\mathcal{A}}$) feature channels of these vertices. Fig. 3.(1) illustrates this process. Here, assume the vertex index $e = 2$, the convolution filter size $m = 3$, and we focus on the 2-nd aligned vertex $\bar{X}_{2,:}$ of $\bar{X}$. The convolution filter $\sum_{s=1}^{c}(\sum_{j=1}^{m} W_j^{h,s}\bar{X}_{2+j-1,s})$ represented by the red lines first maps the s-th feature channels of the 2-nd aligned vertex $\bar{X}_{2,:}$ as well as its neighbor vertices $\bar{X}_{3,:}$ and $\bar{X}_{4,:}$ into a new single value by $\sum_{j=1}^{m} W_j^{h,s}\bar{X}_{2+j-1,s}$, and then sums up the values computed through all the channels as the h-th feature channel of $Z_{2,:}$. Moreover, we need to slide the convolution filter over all the aligned vertices, and this requires three convolution filters represented by the green, red and blue lines respectively. The weights for the three filters are shared, i.e., they are in fact the same filter. Finally, the second step $\sigma(\bar{\mathcal{X}}_h + b^h)$, where $\bar{\mathcal{X}}_h := \sum_{s=1}^{c}(\sum_{j=1}^{m} W_j^{h,s}X_{e+j-1,s})$, applies the Relu function associated with the bias b^h and outputs the final result as $Z_{e,h}$.

To further extract the multi-scale features for a graph associated with its aligned vertex-based grid structure $\bar{X}$, we stack multiple vertex convolution layers defined as follows

$$Z_{e,h}^{t} = \sigma\Big(\sum_{s=1}^{c}\Big(\sum_{j=1}^{m} W_j^{t,h,s}Z_{e+j-1,s}^{t-1}\Big) + b^{t,h}\Big), \tag{8}$$

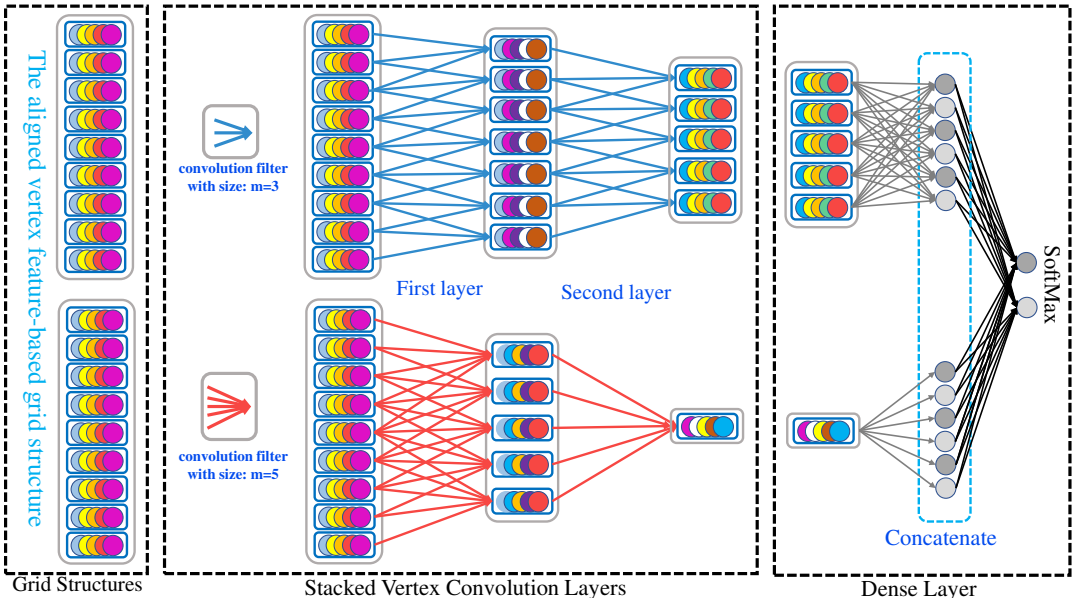


Fig. 4. An example of the General ACVN architecture.

where t is the stack label, Z^0 is the input aligned vertex-based grid structure $\bar{X}_p^{\mathcal{F}} \in \mathbb{R}^{M\times c}$ or $\bar{X}_p^{\mathcal{A}} \in \mathbb{R}^{M\times M}$, and the corresponding notations of the symbols are listed in Table I. After a number of vertex convolution operations, we can employ the Softmax layer to read the extracted features computed from the vertex convolution layers and predict the graph class for graph classifications.

For the vertex convolution operation defined by Eq.(7) and Eq.(8), since the spatial positions of the aligned vertices residing on the required aligned vertex-based grid structure are indexed by the prototype representations, that represent the main characteristics of the aligned vertices and are rearranged based on their interior global similarities (see Section III-B for details). The proposed vertex convolution operation can be seen as the process to gradually aggregate the similar local-level aligned vertices as a new extracted aligned vertex.

B. The Architectures of the AVCN Model

In this subsection, we define the architectures of the proposed AVCN models associated with the new aligned vertex convolution operation defined in Section IV-A. We propose two variants of the AVCN model, namely a) the General ACVN model based on the aligned vertex feature-based grid structure, together with b) the Hybrid AVCN model based on the aligned vertex features and also the aligned vertex adjacency grid structures. We apply these two architectures to graph classification problems.

The General AVCN model: For the General AVCN model, we commence by converting each graph $G(V,E) \in \mathbf{G}$ into the fixed-sized aligned vertex feature-based grid structure $\bar{X}_p^{\mathcal{F}}$. To extract different hierarchical multi-scale feature representations for the aligned vertices, we input the grid structure $\bar{X}_p^{\mathcal{F}}$ of each graph G_p to a family of parallel stacked vertex convolution layers associated with different convolution filter sizes. The architecture of the General AVCN model consists of three convolution layers and is defined as

$$C_{\mathcal{F}:k}^{1:f:(s_1;s_2;\ldots;s_f)} - C_{\mathcal{F}:k}^{2:f:(s_1;s_2;\ldots;s_f)} - C_{\mathcal{F}:k}^{3:f:(s_1;s_2;\ldots;s_f)} - F_u^f, \tag{9}$$

where $C_{\mathcal{F}:k}^{t:f:(s_1;s_2;\ldots;s_f)}$ denotes the t-th ($t = 1$, 2 or 3) vertex convolution layer consisting of f parallel vertex convolution filters each with k channels. The filter sizes of each layer are s_1, s_2, …, s_f respectively and satisfy $s_1 < s_2 < \ldots < s_f$. The subscript $\mathcal{F}$ of $C_{\mathcal{F}:k}^{l:f:(s_1;s_2;\ldots;s_f)}$ indicates that the convolution operation is based on the aligned vertex feature-based grid structure $\bar{X}_p^{\mathcal{F}}$. Finally, F_u^f denotes the dense layer consisting of f parallel fully-connected layers each with u hidden units, where each full-connected layer is added after a corresponding convolution filter of the last stacked convolution layer. An example of the architecture

$$C_{\mathcal{F}:5}^{1:2:(3;5)} - C_{\mathcal{F}:5}^{2:2:(3;5)} - F_6^2$$

for the proposed General AVCN model is shown in Fig. 4. Here, each t-th vertex convolution layer $C_{\mathcal{F}:5}^{t:2:(3;5)}$ ($t = 1$ or 2) has two parallel convolution filters of sizes 3 and 5, the number of channels for each filter is 5, and the stride of each filter is 1. With the extracted patterns learned from the parallel stacked vertex convolution layers to hand, we add the dense layer F_6^2 consisting with 2 parallel fully-connected layers after the final vertex convolution layer. Finally, a Softmax layer is added after the dense layer to learn the graph class. □

Note that, since the convolution operation of the General AVCN model is based on the aligned vertex feature-based grid structure $\bar{X}_p^{\mathcal{F}}$ that only encapsulates the vertex feature information. Unlike the existing DGCNN and ASGCN models that can propagate the vertex feature information through the vertex adjacency matrix during the convolution operation, the proposed General AVCN model can not reflect the topological information residing on the adjacency matrix. To address this shortcoming, we propose a Hybrid AVCN model as follows.

The Hybrid AVCN model: For the proposed Hybrid AVCN model, we commence by converting each graph $G(V,E) \in \mathbf{G}$ into the aligned vertex feature-based grid structure $\bar{X}_p^{\mathcal{F}}$ as

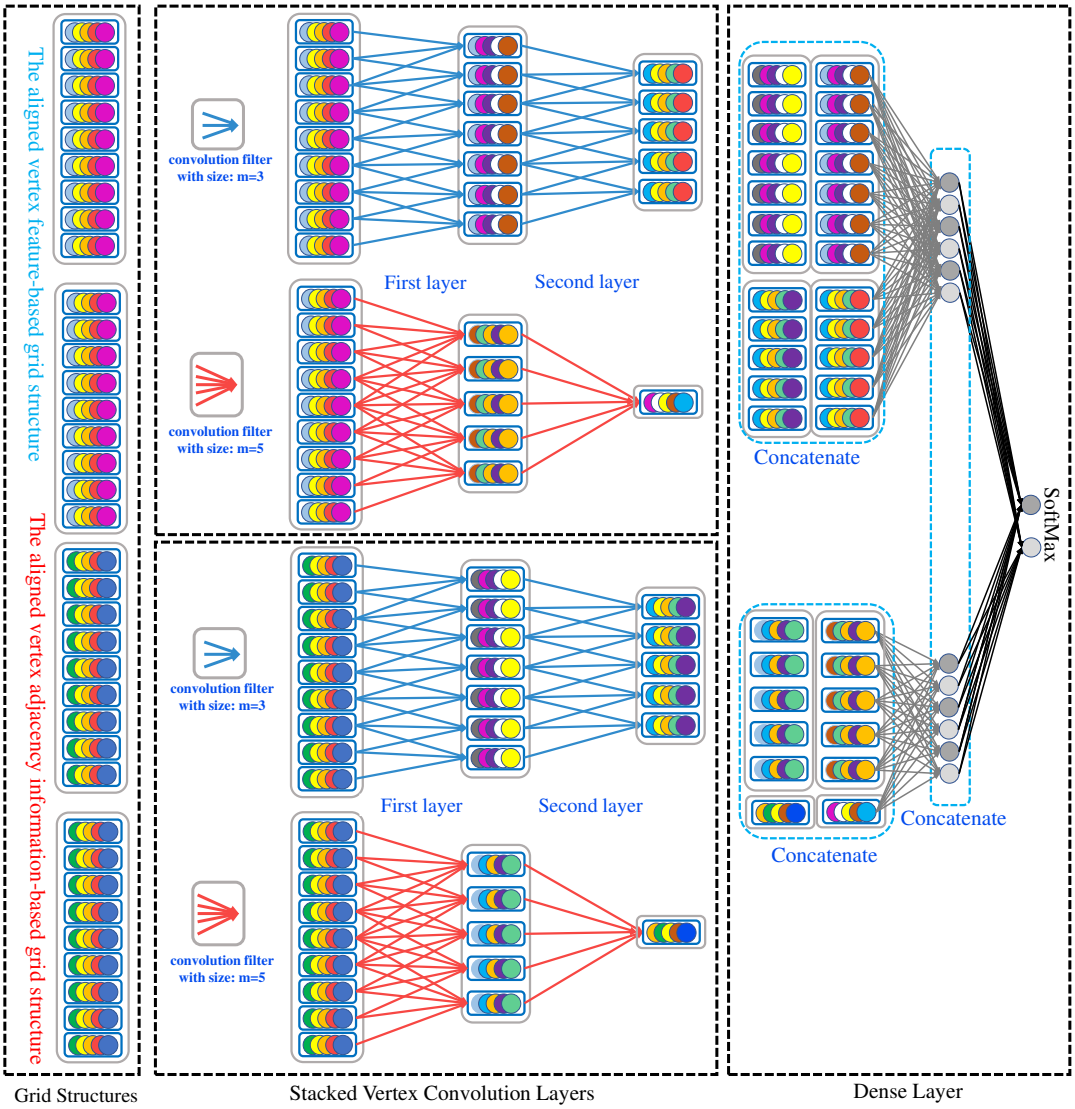


Fig. 5. An example of the Hybrid ACVN architecture.

well as the aligned vertex adjacency information-based grid structure $\bar{X}_p^{\mathcal{A}}$. To extract different hierarchical multi-scale representations, for each graph G_p, we input its grid structures $\bar{X}_p^{\mathcal{F}}$ and $\bar{X}_p^{\mathcal{A}}$ into two families of parallel stacked vertex convolution layers, respectively. The architecture of the Hybrid AVCN model consists of three convolution layers and is defined as

$$\left\{ \begin{array}{l} C_{\mathcal{F}:k}^{1:f:(s_1;s_2;...;s_f)} - C_{\mathcal{F}:k}^{2:f:(s_1;s_2;...;s_f)} - C_{\mathcal{F}:k}^{3:f:(s_1;s_2;...;s_f)} \\ C_{\mathcal{A}:k}^{1:f:(s_1;s_2;...;s_f)} - C_{\mathcal{A}:k}^{2:f:(s_1;s_2;...;s_f)} - C_{\mathcal{A}:k}^{3:f:(s_1;s_2;...;s_f)} \end{array} \right\} F_u^f, \tag{10}$$

where each t-th ($t = 1$, 2 or 3) vertex convolution layer $C_{\mathcal{F}:k}^{t:f:(s_1;s_2;...;s_f)}$ or $C_{\mathcal{A}:k}^{t:f:(s_1;s_2;...;s_f)}$ follows the same definition of the General AVCN model defined by Eq.(9). As a result, the Hybrid AVCN model can be seen as the model consisting of two individual General AVCN models associated with the grid structures $\bar{X}_p^{\mathcal{F}}$ and $\bar{X}_p^{\mathcal{A}}$, respectively. However, unlike the General AVCN model, after the last stacked convolution layers, we propose to first concatenate the extracted patterns from the same sized convolution filters of all different stacked convolution layers from the two General AVCN models as f sets of concatenated features, and then add the predefined dense layer F_u^f associated with f parallel fully-connected layers. An example of the architecture

$$\left\{ \begin{array}{l} C_{\mathcal{F}:5}^{1:2:(3;5)} - C_{\mathcal{F}:5}^{2:2:(3;5)} \\ C_{\mathcal{A}:5}^{1:2:(3;5)} - C_{\mathcal{A}:5}^{2:2:(3;5)} \end{array} \right\} F_6^2$$

for the proposed Hybrid AVCN model is shown in Fig. 5. Unlike the General AVCN model, the Hybrid AVCN model can simultaneously capture either the original vertex feature information or the topological information. □

C. Discussions of the AVCN Model

The proposed Aligned Vertex Convolution Network (AVCN) model has a number of novelties and advantages, that are not available for most existing state-of-the-art GCN models. These are listed in Table II and discussed as follows.

First, rather than pooling vertex features, we aggregate them with an aligned multi-scale grid structure. The Neural Graph Fingerprint Network (NGFN) model [17] as well as the Diffusion Convolution Neural Network (DCNN) model [18] both employ a SumPooling layer to directly compute the summation of the local-level features of vertices abstracted

TABLE II
PROPERTIES OF DIFFERENT GCN MODELS.

Properties	AVCN	NGFN [17]	DCNN [18]	PSGCNN [19]	DGCNN [21]	ASGCN [22]
Focus More on Local Information	**Yes**	No	No	**Yes**	**Yes**	**Yes**
Encapsulate Structural Correspondence Information	**Yes**	No	No	No	No	**Yes**
Restrict Over-smoothing Problem	**Yes**	No	No	**Yes**	No	No
Preserve All Original Vertex Information	**Yes**	**Yes**	**Yes**	No	No	**Yes**
Discriminate Importance between Different Vertices	**Yes**	No	No	**Yes**	No	**Yes**

from the convolution layers as the global-level characteristics of graph structures. By contrast, the proposed AVCN model focuses more on learning local structural features through the proposed aligned vertex-based grid structure. Specifically, Fig. 3 implies that the associated vertex convolution operation of the proposed AVCN model can convert the original aligned vertex-based grid structure into a new packed grid structure, by packing the aligned vertex features from the original grid structure into the new grid structure. Thus, the new grid structure can be viewed as a new extracted aligned vertex-based grid structure with a reduced number of aligned vertices. As a result, the proposed AVCN model can gradually extract multi-scale local-level vertex features through a number of stacked vertex convolution layers, and encapsulate more significant local-level structural information than the existing DCNN and NGFN models based on SumPooling.

Second, our model ensures both consistent spatial and consistent structural alignment of vertex features. Like the proposed AVCN model, either the PATCHY-SAN Graph Convolution Neural Network (PSGCNN) model [19] or the Deep Graph Convolution Neural Network (DGCNN) model [21] needs to predetermine the vertex orders of each graph and convert the graph into the fixed-sized vertex grid structure. Unfortunately, both the PSGCNN and the DGCNN models sort the vertex orders based on each individual graph structure, ignoring the arrangement of consistent vertex correspondence information between different graphs. By contrast, the proposed AVCN model employs a transitive vertex matching method to convert arbitrary sized graphs into fixed-sized aligned vertex-based grid structures where the aligned vertices on the same spatial position are also structurally aligned. Moreover, the proposed AVCN can keep the consistency between the spatial positions and structural correspondences for the abstracted grid structures after the vertex convolution operation. Thus, our AVCN model can always encapsulate the structural correspondence information over all graphs during the computational process of convolution operations.

Third, unlike the Aligned-Spatial Graph Convolution Network (ASGCN) model [22] and the DGCNN model, our proposed AVCN model can avoid over-smoothing vertex features through a process of gradually multiscale aggregation. By contrast, the ASGCN and DGCNN models rely on propagating feature information between adjacent vertices, and in turn produce similar abstracted vertex features and fail to capture local vertex information. On the other hand, as discussed previously, the DGCNN and PSGCNN models also need to construct fixed-sized vertex grid structures for graph classification. Both methods may discard the vertices with lower ranking during the construction process. Unlike the DGCNN and PSGCNN models, the aligned vertex-based grid structures of our AVCN model can preserve the original vertex features and vertex adjacency information from original graphs. Thus, our AVCN models address the drawback of information loss arising in the DGCNN and PSGCNN models.

Fourth, our model does not unnecessarily discard vertex features. Similar to the proposed AVCN model, the ASGCN model is also defined based on the fixed-sized aligned vertex-based grid structure. Thus, ASGCN model can also either reduce the information loss or overcome the neglect of structural correspondence information arising in most existing spatial-based GCN models. However, like the DGCNN model, the associated convolution operation of the ASGCN model depends on propagating the feature information between neighboring vertices indicated by the vertex adjacency matrix and this process may propagate redundant feature information between any pair of adjacent vertices for multiple times. As a result, both the ASGCN and the DGCNN model suffer from the notorious over-smoothing problem with multiple graph convolution layers [6]. By contrast, the required vertex convolution operation of the AVCN model can gradually aggregate the neighboring local-level aligned vertices residing on the original grid structures as a new packed aligned vertex, i.e., the proposed AVCN model can avoid iteratively propagating redundant information between specific neighboring vertices during the convolution operation process. Thus, the AVCN model can significantly restrict the over-smoothing problem.

Fifth, our model can adaptively discriminate the importance of vertices. the required vertex convolution operation of the proposed AVCN model can be seen as an one-dimensional standard convolution filter of the CNN on standard grid structures [1]. The AVCN model can thus assign the neighboring aligned vertices a family of different parameter weights during the convolution operation process. As a result, similar to the ASGCN and PSGCNN models, the AVCN model can also adaptively discriminate the importance of different vertices. In this way it addresses the problem of ignoring the vertex importance information, which arises in the spatial-based NGFN, DCNN and DGCNN models.

Finally, the aligned grid structures of the AVCN model are constructed by transitively aligning each original graph to prototype representations that are identified by the classical k-means clustering method. Although, the k-means method needs to randomly select initial centers from the original vectorial DB representations of vertices over all graphs. This does not influence the robustness of the resulting grid structures. Because the associated transitive vertex alignment method is based on the k-dimensional ($k \leq K$) DB representations and we need to perform the k-means method for K times to construct the resulting grid structures.

V. EXPERIMENTS

In this subsection, we compare the performance of the proposed AVCN models to state-of-the-art approaches on graph classification problems with ten standard open source graph

datasets [31]. These datasets are extracted from bioinformatics, computer vision and social networks, respectively. Statistical information of these datasets are exhibited in Table III.

A. Evaluations on Graph Classification

Experimental Setup: We evaluate the performance of our AVCN models on graph classification problems, including the AVCN(G) model with the general AVCN architecture and the AVCN(H) model with the hybrid architecture stated in Section IV-B. Moreover, we compare our model against alternative graph kernels and deep learning methods for graphs. Specifically, the graph kernels include: 1) the Weisfeiler-Lehman Subtree Kernel (WLSK) [32], 2) the Weisfeiler-Lehman Kernel associated with Core Variants (CORE WL) [33], 3) Jensen-Tsallis q-difference Kernel (JTQK) with $q = 2$ [34], 4) the Shortest Path Graph Kernel (SPGK) [35], 5) the Shortest Path Kernel associated with Core Variants (CORE SP) [33], 6) the Random Walk Graph Kernel (RWGK) [36], 7) the Graphlet Kernel (GK) [37], and 8) the Pyramid Match Graph Kernel (PMGK) [38]. On the other hand, the deep learning methods include: 1) the Aligned-Spatial Graph Convolution Network (ASGCN) mdoel [22], 2) the backtrackless version of the ASGCN model (i.e., the BASGCN model [23]), 3) the PATCHY-SAN based Convolutional Neural Network (PSGCNN) [19], 4) the Deep Graph Convolutional Neural Network (DGCNN) [21], 5) the Diffusion Convolutional Neural Network (DCNN) [18], 6) the Anonymous Walk Embeddings with Feature Driven (AWE) [39], 7) the Deep Graphlet Kernel (DGK) [40], 8) the Self-Attention Pooling based Graph Convolution Network (SAGPool) [41], 9) the Differentiable Pooling based Graph Convolution Network (DiffPool) [42], 10) the EigenPooling based Graph Convolution Network (EigenPool) [43], 11) the Degree-specific Graph Neural Network (DEMO-Net) [44], 12) the Edge-conditioned Convolutional Network (ECC) [45], and 13) the High-order Graph Convolution Network (HO-GCN) [46].

For the experiment, we propose to use the same network structure for either the AVCN(G) model or the AVCN(H) model on all graph datasets. The reasons of adopting the same structure are twofold. First, utilizing the same GCN network structure can guarantee the fair comparison between the proposed AVCN models and the graph kernel methods [21]. Second, it is useful to evaluate the generalization ability of the proposed AVCN models on different datasets [23]. Specifically, for either the AVCN(G) or the AVCN(H) model, we set the prototype representation number as $M = 64$. This is because we observe that the vertex numbers of most graphs (about 60% to 70% graphs) over all datasets are around 64. This setting in turn guarantees that the required aligned vertex-based grid structures can preserve the vertex feature and adjacency matrix of original graphs as much as possible. Moreover, this setting can also guarantee the better trade-off between the classification performance and the computational efficiency, because greater parameter M will lead to larger network structures for the proposed AVCN model. We input the grid structures into the AVCN(G) or the AVCN(H) model associated with three parallel stacked vertex convolution layers, where each layer has four parallel vertex convolution filters of sizes 3, 5, 7 and 9, respectively. Moreover, we set the number of channels for each vertex convolution filter as 64, the stride number for each filter as 1, and the number of the hidden units for the final fully-connected layer as 128. As a result, based on Eq.(9) and Eq.(10), the resulting architectures of the AVCN(G) and AVCN(H) models are

$$C_{\mathcal{F}:64}^{1:4:(3;5;7;9)} - C_{\mathcal{F}:64}^{2:4:(3;5;7;9)} - C_{\mathcal{F}:64}^{3:4:(3;5;7;9)} - F_{128}^{4},$$

and

$$\left\{ \begin{array}{l} C_{\mathcal{F}:64}^{1:4:(3;5;7;9)} - C_{\mathcal{F}:64}^{2:4:(3;5;7;9)} - C_{\mathcal{F}:64}^{3:4:(3;5;7;9)} \\ C_{\mathcal{A}:64}^{1:4:(3;5;7;9)} - C_{\mathcal{A}:64}^{2:4:(3;5;7;9)} - C_{\mathcal{A}:64}^{3:4:(3;5;7;9)} \end{array} \right\} F_{128}^{4},$$

respectively. Note that, for both the AVCN(G) model and the AVCN(H) model, we also add a classical AvgPooling layer of size and stride 2 after the first and second stacked vertex convolution layers. We concatenate the features from their fully-connected layer, and add a Softmax layer with a dropout rate of 0.5 by following the same setting of existing works [21]. We use the rectified linear units (ReLU) for their convolution filters. The only optimized hyperparameters are a) the learning rate, b) the number of epochs, and c) the batch size for the mini-batch gradient decent algorithm.

Note that, the proposed AVCN models need to first construct the prototype representations to identify the transitive correspondence information between vertices over all graphs. We propose to abstract the prototype representations ($\mathbf{PR}$s) from either the training or testing graphs. Hence, the proposed AVCN models can be viewed as the instance of transductive learning [47], where all graphs are employed to abstract the prototype representations. However, note that, computing the $\mathbf{PR}$s does not use any label information from both training and testing data. Moreover, the $\mathbf{PR}$s are only employed to map each graph into the fixed-sized grid structure, and the training process of the proposed AVCN model does not use any testing graph information. As a result, the train process of either the AVCN(G) model and the AVCN(H) model is still inductive. In fact, the alternative deep learning models as well as the graph kernels for attributed graphs in our comparisons can also be seen as instances of transductive learning. This is because these methods can accommodate vertex labels. They thus need to seek the label space over all the training and testing graphs for constructing one hot coding vertex feature matrix (for deep learning methods) or initializing the vertex label (for kernels). However, similar to the proposed model, the training processes for these alternative methods are only based on the training graphs, i.e., their processes are still inductive.

For our proposed AVCN models, we employ 10-fold cross-validation to calculate the mean classification accuracy for each dataset, where we use 9 sample sets for training and 1 sample set for testing. For each of the datasets, we perform the experiments 10 times and show the mean classification accuracy as well as the standard error in Table IV. In terms of the alternative kernel methods, we set the parameters of the maximum subtree height for both the WLSK and JTQK kernels as 10, based on the previous empirical studies in the original papers. For each alternative graph kernel, we employ

TABLE III
INFORMATION OF THE GRAPH DATASETS

Datasets	MUTAG	PROTEINS	D&D	PTC(MR)	GatorBait	Reeb	IMDB-B	IMDB-M	RED-B	COLLAB
Max # vertices	28	620	5748	109	545	220	136	89	3782	492
Mean # vertices	17.93	39.06	284.32	25.56	348.72	95.42	19.77	13.00	429.62	74.49
Mean # eges	19.79	72.82	715.65	25.96	796.11	94.59	96.53	65.93	497.75	2457.50
# graphs	188	1113	1178	344	100	300	1000	1500	2000	2000
# vertex labels	7	3	82	19	78	32	–	–	–	–
# classes	2	2	2	2	30	20	2	3	2	2
Description	BioInfor	BioInfor	BioInfor	BioInfor	CV	CV	Social	Social	Social	Social

TABLE IV
PERFORMANCE COMPARISONS WITH GRAPH KERNELS.

Datasets	MUTAG	PROTEINS	D&D	PTC(MR)	GatorBait
AVCN(G)	87.05 ± 0.71	**75.71** ± 0.65	**80.10** ± 0.95	**60.13** ± 0.70	**19.00** ± 0.75
AVCN(H)	**89.30** ± 0.63	**75.75** ± 0.43	**80.77** ± 0.77	**62.32** ± 0.67	**22.90** ± 1.07
JTQK	85.50 ± 0.55	72.86 ± 0.41	79.89 ± 0.32	58.50 ± 0.39	11.40 ± 0.52
WLSK	82.88 ± 0.57	73.52 ± 0.43	79.78 ± 0.36	58.26 ± 0.47	10.10 ± 0.61
CORE WL	87.47 ± 1.08	–	79.24 ± 0.34	59.43 ± 1.20	–
SPGK	83.38 ± 0.81	75.10 ± 0.50	78.45 ± 0.26	55.52 ± 0.46	9.00 ± 0.75
CORE SP	88.29 ± 1.55	–	77.30 ± 0.80	59.06 ± 0.93	–
PMGK	80.66 ± 0.90	–	77.34 ± 0.97	56.41 ± 1.45	–
GK	81.66 ± 2.11	71.67 ± 0.55	78.45 ± 0.26	52.26 ± 1.41	$8.40 \pm .83$
RWGK	80.77 ± 0.72	74.20 ± 0.40	71.70 ± 0.47	55.91 ± 0.37	7.00 ± 0.77

Datasets	Reeb	IMDB-B	IMDB-M	RED-B	COLLAB
AVCN(G)	**67.00** ± 0.91	72.75 ± 0.39	51.19 ± 0.49	90.50 ± 0.20	**79.12** ± 0.25
AVCN(H)	**70.20** ± 0.68	73.46 ± 0.59	50.90 ± 0.35	**91.22** ± 0.36	**80.24** ± 0.26
JTQK	60.56 ± 0.35	72.45 ± 0.81	50.33 ± 0.49	77.60 ± 0.35	76.85 ± 0.40
WLSK	58.53 ± 0.53	71.88 ± 0.77	49.50 ± 0.49	76.56 ± 0.30	77.39 ± 0.35
CORE WL	–	**74.02** ± 0.42	**51.35** ± 0.48	78.02 ± 0.23	–
SPGK	55.73 ± 0.44	71.26 ± 1.04	51.33 ± 0.57	84.20 ± 0.70	58.80 ± 0.20
CORE SP	–	72.62 ± 0.59	49.43 ± 0.42	90.84 ± 0.14	–
PMGK	81.66 ± 2.11	71.67 ± 0.55	78.45 ± 0.26	52.26 ± 1.41	$8.40 \pm .83$
GK	–	68.53 ± 0.61	45.75 ± 0.66	82.70 ± 0.68	–
RWGK	32.47 ± 0.69	67.94 ± 0.77	46.72 ± 0.30	72.73 ± 0.39	–

TABLE V
PERFORMANCE COMPARISONS WITH DEEP LEARNING APPROACHES ON BIOINFORMATICS AND SOCIAL NETWORK DATASETS..

Datasets	MUTAG	PROTEINS	D&D	PTC(MR)	IMDB-B	IMDB-M	RED-B	COLLAB
AVCN(G)	87.05 ± 0.71	75.71 ± 0.65	80.10 ± 0.95	60.13 ± 0.70	72.75 ± 0.39	51.19 ± 0.49	90.50 ± 0.20	79.12 ± 0.25
AVCN(H)	89.30 ± 0.63	75.75 ± 0.43	**80.77** ± 0.77	**62.32** ± 0.67	73.46 ± 0.59	50.90 ± 0.35	**91.22** ± 0.36	**80.24** ± 0.26
ASGCN	89.70 ± 0.85	**76.50** ± 0.59	80.40 ± 0.95	61.42 ± 2.47	73.86 ± 0.92	50.86 ± 0.85	90.60 ± 0.35	78.75 ± 0.79
BASGCN	**90.05** ± 0.82	76.05 ± 0.57	80.71 ± 0.99	61.51 ± 0.77	74.00 ± 0.87	$50.43 \pm .77$	91.00 ± 0.25	79.60 ± 0.83
DGCNN	85.83 ± 1.66	75.54 ± 0.94	79.37 ± 0.94	58.59 ± 2.47	70.03 ± 0.86	47.83 ± 0.85	76.02 ± 1.73	73.76 ± 0.49
PSGCNN	88.95 ± 4.37	75.00 ± 2.51	76.27 ± 2.64	62.29	71.00 ± 2.29	45.23 ± 2.84	86.30 ± 1.58	72.60 ± 2.15
DCNN	66.98	61.29 ± 1.60	58.09 ± 0.53	58.09 ± 0.53	49.06 ± 1.37	33.49 ± 1.42	–	52.11 ± 0.71
DGK	82.66 ± 1.45	71.68 ± 0.50	78.50 ± 0.22	57.32 ± 1.13	66.96 ± 0.56	44.55 ± 0.52	78.30 ± 0.30	73.09 ± 0.25
AWE	87.87 ± 9.76	–	71.51 ± 4.02	–	73.13 ± 3.28	**51.58** ± 4.66	82.97 ± 2.86	70.99 ± 1.49
HO-GCN	86.10	–	75.50	60.90	**74.20**	49.50	–	–

10-fold cross-validation associated with the LIBSVM of C-Support Vector Machines (C-SVMs) to calculate the mean classification accuracy. We perform the experiments 10 times for each kernel on each dataset, and show the mean classification accuracy as well as the standard error in Table IV. Because some kernels have been well evaluated by other authors based on the same setting of ours, we directly exhibit the corresponding results of these kernels from the original literatures. Note that, the symbol – in Table IV indicates that some approaches were not evaluated on the corresponding datasets by the original authors, and this symbol has the same meaning in the following Table V and Table VI

On the other hand, in terms of the alternative deep learning approaches, we show the best results for the ASGCN, BASGCN, DGCNN, PSGCNN, HO-GCN and DGK models reported in their original publications. For the DCNN model, we directly show the results from the work of Zhang et al., [21], associated with the same experimental setting as our methods. For the AWE model, we show the classification accuracies based on the feature-driven AWE, because of its better performance on attributed graphs. Note that, the PSGCNN model is able to leverage extra edge features, whereas most alternative methods can not leverage these features.

TABLE VI
PERFORMANCE COMPARISONS WITH DEEP LEARNING APPROACHES ON BIOINFORMATICS DATASETS.

Datasets	MUTAG	PROTEINS	D&D	PTC(MR)
AVCN(G)	**87.05**	75.71	80.10	**60.13**
AVCN(H)	**89.30**	75.75	**80.77**	**62.32**
ECC	76.11	–	72.54	–
DEMO-Net	81.40	–	70.80	57.20
DiffPool	82.66	76.25	80.64	–
EigenPool	79.50	**78.60**	76.60	–
SAGPool	–	71.86	76.45	–

Hence, for the PSGCNN model, we only show the results based on vertex features. Moreover, as these alternative deep learning approaches have not been investigated on the Reeb and GatorBait datasets abstracted from computer vision by any author, we do not include the accuracies for these methods. The classification accuracies associated with standard errors of each deep learning approach are listed in Table V.

Finally, the DEMO-Net, EigenPool and SAGPool models were not investigated on the social network datasets in the original publications. Both the DiffPool and ECC models, on the other hand, were only evaluated on the COLLAB dataset in the original publications, and their accuracies are 67.79 and 75.48, respectively. These are obviously lower than the proposed models. Thus, we only show the mean accuracies of these models on the bioinformatics datasets in Table VI.

Experimental Results: Table IV, Table V and Table VI indicate that the proposed AVCN models can outperform either the graph kernels or the deep learning approaches for graphs, on most datasets. Specifically, in terms of the comparisons with the graph kernels on the standard bioinformatics, computer vision and social network datasets, Table IV indicates that the proposed AVCN models can achieve better performance than the alternative graph kernels on eight of the ten datasets. By contrast, each of the alternative methods can only achieve the best classification accuracies on at most two of the ten datasets. Although the classification accuracies of the proposed AVCN models on the IMDB-B as well as IMDB-M datasets are not the best, our AVCN models are still competitive and outperform most of the graph kernels. In terms of the comparisons with the deep learning approaches on the standard social network and bioinformatics datasets, Table V indicates that the proposed AVCN models can outperform the alternative deep learning approaches on four of the eight datasets. By contrast, each of the alternative methods can only achieve the best classification accuracies on at most one of the eight datasets. Although the classification accuracies of the proposed AVCN models on the MUTAG, PROTEINS, IMDB-B and IMDB-M datasets are not the best, our models are still competitive and outperform most of the deep learning methods. Overall, the ASGCN as well as the BASGCN models are the most competitive alternative methods when compared with the proposed AVCN model. However, our model can still outperform these two alternative models on five of the eight datasets. In terms of the comparisons with the partial GCN models on the standard bioinformatics datasets, Table VI indicates that our proposed methods can outperform all of the alternative GCN models, but excluding the EigenPool model on the PROTEINS datasets.

Experimental Analysis: In general, although several alternative approaches may achieve better classification accuracies than the proposed AVCN models on a small number of datasets, our models are still competitive on these datasets, and outperform these methods on most of the remaining datasets. The experimental results indicate the effectiveness of the proposed AVCN model. Moreover, although both the AVCN(G) and AVCN(H) models are effective, the performance of the AVCN(H) is obviously better than that of the AVCN(G) model. This is because the AVCN(G) model can only accommodate the aligned vertex feature-based grid structures. By contrast, the AVCN(H) model can accommodate both the aligned vertex features and the adjacency-based grid structures. Thus, only the AVCN(H) model can simultaneously capture both the vertex feature information and the topological information of graphs. Overall, the reasons for the effectiveness of the proposed AVCN models are fourfold.

First, the alternative graph kernels are typical examples of R-convolution kernels and are based on measuring the similarity of substructures, without using the correspondence information. By contrast, the proposed AVCN model associates the aligned vertex-based grid structures, that incorporates the transitive vertex alignment information between graphs, and thus better reflects graph characteristics. Furthermore, the C-SVMs associated with kernel methods can only be viewed as a classical framework of shallow learning [48]. In contrast, our AVCN models offer an end-to-end framework of deep learning, extracting more meaningful characteristics of graphs.

Second, similar to the alternative graph kernels, all the alternative deep learning approaches fail to integrate the correspondence information between graph structures into their learning frameworks, excluding the alternative ASGCN and BASGCN models. In particular, either the DGCNN model or the PASGCNN model needs to rearrange the vertex orders and some vertices may be lost. Clearly, this may cause significant information loss. In contrast, our AVCN models are able to encapsulate more information from the original graphs.

Third, unlike the proposed AVCN models, some alternative spatial-based GCN models (e.g., the DCNN model) need to compute the summation of extracted local-level features of vertices as global-level characteristics of graph structures. In contrast, the proposed AVCN models can learn richer multi-scale local-level vertex features. Experiments demonstrate the effectiveness of the proposed models.

Fourth, although the ASGCN and BASGCN models are also designed based on aligned vertex-based grid structures, and can thus reflect the structural correspondence information and reduce the information loss problem like the proposed AVCN model. Unfortunately, like the DGCNN model, the ASGCN and BASGCN models also suffer from over-smoothing. By contrast, the proposed AVCN model can significantly restrict this drawback, and thus extract more discriminating multi-scale features for graph classification.

Finally, the proposed AVCN models mainly have a little lower classification performance than the ASGCN and BASGCN models on the MUTAG, PROTEINS, IMEB-B datasets. Through Table III, we observe that the three datasets have obviously less average vertex numbers than most of the remaining datasets. Since the proposed AVCN models maily rely on capturing local-level vertex information, this may in turn influence the performance of the AVCN models. However, the proposed AVCN models are still competitive on these datasets, demonstrating the effectiveness.

B. Evaluation of Different Prototype Representation Numbers

To further analyze the performance of the proposed AVCN models, in this subsection we investigate how the selection of the parameter M (i.e., the numbers of the prototype representations) affects the classification performance with the proposed AVCN(H) model. As we have stated previously in Section V-A, the sizes of 60% to 70% graphs over all the datasets are around 64. Setting the parameter M as 64 can not only preserve the structural information of most original graphs as much as possible, but also guarantee the better trade-off between the classification performance and the computational efficiency. Thus, in this experiment, we vary the parameter M from 16 to 64 with 8 strides, and exhibit how the classification performance of the proposed AVCN(H) varies with the increasing parameter M. Specifically, we only perform the AVCN(H) model on the RED-B, PTC and COLLAB datasets, due to the representativeness of different levels of average graph sizes, i.e., the graph size is around 64

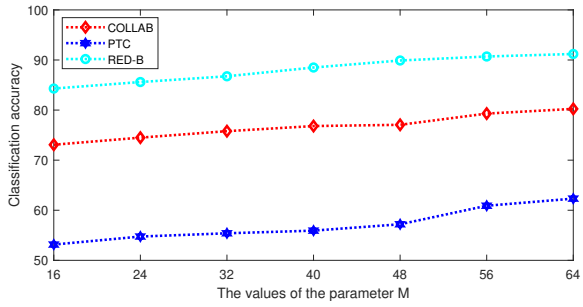


Fig. 6. Accuracies vs different parameters M.

for the COLLAB dataset, a little larger than 64 for the PTC dataset, and much greater than 64 for the RED-B dataset. The experimental results are shown in Fig.6. This figure indicates that the classification performance of the our AVCN(H) model tends to gradually increase with M until it reaches a plateaux when M is greater than 48 or 56. This is because the grid structure sizes of the proposed AVCN models are related to the value of the parameter M, and the greater value of M can preserve more structural information of original graphs, influencing the classification performance of the proposed AVCN model.

VI. Conclusion

In this work, we have developed a family of Aligned Vertex Convolutional Network (AVCN) models for graph classification. Our approaches are based on employing a transitive vertex matching method to convert the graphs of arbitrary sizes into fixed-sized aligned vertex-based grid structures, and then designing a new aligned vertex convolution operation on the associated grid structures. Since the proposed vertex convolution operation can gradually aggregating local-level neighboring aligned vertices residing on the original grid structures as a new packed aligned vertex, i.e., the convolution operation can extract new packed grid structures with a reduced number of packed aligned vertices. The proposed AVCN models can avoid iteratively propagating redundant information between specific neighboring vertices, and significantly restrict the notorious over-smoothing problem arising in most spatial-based GCN models. Experimental evaluations on benchmark datasets demonstrate the effectiveness.

Acknowledgments

We thank Mr. Yuhang Jiao's help on partial coding and experimental works, as well as Dr. Shu Wu's suggestions on this paper.

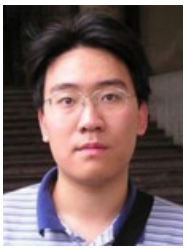
References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[2] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, 2016.
[3] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, 2016.
[4] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.
[5] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *Proceedings of ICML*, 2019, pp. 7134–7143.
[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of ICLR*, 2017.
[7] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of AAAI*.
[8] X. Zhang, C. Xu, X. Tian, and D. Tao, "Graph edge convolutional neural networks for skeleton-based action recognition," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 8, pp. 3047–3060, 2020.
[9] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognit.*, vol. 97, 2020.
[10] Y. Chen, G. Ma, C. Yuan, B. Li, H. Zhang, F. Wang, and W. Hu, "Graph convolutional network with structure pooling and joint-wise channel attention for action recognition," *Pattern Recognit.*, vol. 103, p. 107321, 2020.
[11] J. Wang, L. Zhang, Q. Wang, L. Chen, J. Shi, X. Chen, Z. Li, and D. Shen, "Multi-class ASD classification based on functional connectivity and functional correlation tensor via multi-source domain adaptation and multi-view sparse representation," *IEEE Trans. Medical Imaging*, vol. 39, no. 10, pp. 3137–3147, 2020.
[12] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *CoRR*, vol. abs/1312.6203, 2013.
[13] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Proceddings of NIPS*, 2015, pp. 2449–2457.
[14] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *CoRR*, vol. abs/1506.05163, 2015. [Online]. Available: http://arxiv.org/abs/1506.05163
[15] X. Bai, E. R. Hancock, and R. C. Wilson, "Graph characteristics from the heat kernel trace," *Pattern Recognit.*, vol. 42, no. 11, pp. 2589–2606, 2009.
[16] 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
[17] 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.
[18] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proceedings of NIPS*, 2016, pp. 1993–2001.
[19] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of ICML*, 2016, pp. 2014–2023.
[20] Z. Zhang, D. Chen, J. Wang, L. Bai, and E. R. Hancock, "Quantum-based subgraph convolutional neural networks," *Pattern Recognit.*, vol. 88, pp. 38–49, 2019.
[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] L. Bai, Y. Jiao, L. Cui, and E. R. Hancock, "Learning aligned-spatial graph convolutional networks for graph classification," in *Proceedings of ECML-PKDD, Part I*, 2019, pp. 464–482.
[23] L. Bai, L. Cui, Y. Jiao, L. Rossi, and E. R. Hancock, "Learning backtrackless aligned-spatial graph convolutional networks for graph classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. In Press, 2020.
[24] I. Spinelli, S. Scardapane, and A. Uncini, "Adaptive propagation graph convolutional network," *CoRR*, vol. abs/2002.10306, 2020.
[25] L. Bai, Y. Jiao, L. Cui, L. Rossi, Y. Wang, P. S. Yu, and E. R. Hancock, "Learning graph convolutional networks based on quantum vertex information propagation," *IEEE Transactions on Knowledge and Data Engineering*, p. In Press, 2021.
[26] D. Cai, "Litekmeans: the fastest matlab implementation of kmeans," *Available at: http://www.zjucadcg.cn/dengcai/Data/Clustering.html*, 2011.
[27] 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.
[28] L. Bai and E. R. Hancock, "Depth-based complexity traces of graphs," *Pattern Recognition*, vol. 47, no. 3, pp. 1172–1186, 2014.
[29] L. Bai, L. Rossi, Z. Zhang, and E. R. Hancock, "An aligned subtree kernel for weighted graphs," in *Proceedings of ICML*, 2015, pp. 30–39.
[30] X. Bai, C. Yan, H. Yang, L. Bai, J. Zhou, and E. R. Hancock, "Adaptive hash retrieval with kernel based similarity," *Pattern Recognit.*, vol. 75, pp. 136–148, 2018.
[31] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016. [Online]. Available: http://graphkernels.cs.tu-dortmund.de

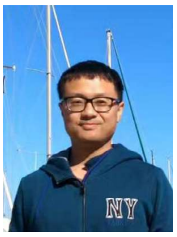
[32] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 1, pp. 1–48, 2010.
[33] G. Nikolentzos, P. Meladianos, S. Limnios, and M. Vazirgiannis, "A degeneracy framework for graph similarity," in *Proceedings of IJCAI*, 2018, pp. 2595–2601.
[34] 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.
[35] 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.
[36] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *Proceedings of ICML*, 2003, pp. 321–328.
[37] 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.
[38] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Proceedings of AAAI*, 2017, pp. 2429–2435.
[39] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," in *Proceedings of ICML*, 2018, pp. 2191–2200.
[40] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 2015, pp. 1365–1374.
[41] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proceedings of ICML*, 2019, pp. 3734–3743.
[42] 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.
[43] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *Proceedings of KDD*, 2019, pp. 723–731.
[44] J. Wu, J. He, and J. Xu, "Demo-net: Degree-specific graph neural networks for node and graph classification," in *Proceedings of KDD*, 2019.
[45] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of CVPR*, 2017, pp. 29–38.
[46] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of AAAI*, 2019.
[47] A. Gammerman, K. S. Azoury, and V. Vapnik, "Learning by transduction," in *Proceedings of UAI*, 1998, pp. 148–155.
[48] 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.

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 has published more than 40 journal and conference papers, including TPAMI, TFS, TNNLS, TKDE, TCYB, PR, IJPR, 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.

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, TKDE, TCYB, TNNLS, PR, ICML, IJCAI, ECML-PKDD, ICDM, etc. His current research interests include pattern recognition, machine learning, quantum walks, and financial data analysis. He is currently a member of the editorial board of the journal Pattern Recognition.



Xiao Bai Xiao Bai received the B.Eng. degree in computer science from Beihang University, Beijing, China, in 2001, and the Ph.D. degree in computer science from the University of York, York, U.K., in 2006. He was a Research Officer (Fellow and Scientist) with the Computer Science Department, University of Bath, Bath, U.K., until 2008. He is currently a Full Professor with the School of Computer Science and Engineering, Beihang University. He has authored or coauthored more than 100 papers in journals and refereed conferences. His current research interests include pattern recognition, image processing, and remote sensing image analysis. He is an Associate Editor of Pattern Recognition and Signal Processing.



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.

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 is the fellow of the Royal Academy of Engineering, and was a recipient of the Royal 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 2007, 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).