Article:
Bai, Lu, Cui, Lixin, Bai, Xiao et al. (1 more author) (2018) Deep depth-based representations of graphs through deep learning networks. Neurocomputing. ISSN 0925-2312

https://doi.org/10.1016/j.neucom.2018.03.087

Deep Depth-based Representations of Graphs through Deep Learning Networks

Lu Bai[1*], Lixin Cui[1*], Xiao Bai[2], Edwin R. Hancock[3]

[1] *Central University of Finance and Economics, Beijing, China.*
[2] *Beihang University, Beijing, China.*
[3] *University of York, York, UK.*

Abstract

Graph-based representations are powerful tools in structural pattern recognition and machine learning. In this paper, we propose a framework of computing the deep depth-based representations for graph structures. Our work links the ideas of graph complexity measures and deep learning networks. Specifically, for a set of graphs, we commence by computing depth-based representations rooted at each vertex as vertex points. In order to identify an informative depth-based representation subset, we employ the well-known k-means method to identify M dominant centroids of the depth-based representation vectors as prototype representations. To overcome the burdensome computation of using depth-based representations for all graphs, we propose to use the prototype representations to train a deep autoencoder network, that is optimized using Stochastic Gradient Descent together with the Deep Belief Network for pretraining. By inputting the depth-based representations of vertices over all graphs to the trained deep network, we compute the deep representation for each vertex. The resulting deep depth-based representation of a graph is computed by averaging the deep representations of its complete set of vertices. We theoretically demonstrate that the deep depth-based representations of graphs not only reflect both the local and global characteristics of graphs through the depth-based representations, but also capture the main structural relationship and information content over all graphs under investigations. Experimental evaluations demonstrate the effectiveness of the proposed method.

[*] These authors contribute equally to this work and are co-first authors

Keywords: Depth-based Representations, Deep Representations, Deep Autoencoder Networks, Prototype Representations

1. Introduction

Analyzing data using graph-based representations has attracted an increasing interest in machine learning and pattern recognition, due to the representational power of graph structures. Typical applications include a) analyzing bioinformatics and chemoinformatics data [1] (e.g., classifying graph-based representations of proteins or chemical compounds into different species), b) recognizing graph-based object descriptions from images [2], c) visualizing social networks [3] (e.g., Twitter and Facebook friendship networks, DBLP citation networks), etc. One challenge arising in analyzing graph-based representations of data is how to convert the discrete graph structures into numeric features, since this allows standard pattern recognition techniques to be directly applied to graphs.

The aim of this paper is to propose a new framework of computing deep depth-based representation of a graph through the use of deep learning networks [4]. Our approach is to identify a family of dominant depth-based representations [5] as prototype representations, and then train a deep network that can better preserve the non-linear graph structure information. The resulting deep depth-based representations of graphs are computed through the trained deep networks.

1.1. Literature Review

Although graph-based representations are powerful tools for structural pattern recognition, the main drawback associated with graph structures is the lack of a correspondence order or labels for the vertices, i.e., we do not explicitly know how to align different graph adjacency matrices. Compared to vector-based pattern recognition, this in turn limits the set of standard machine learning algorithms that can be directly applied to problems of graph classification or clustering [6]. One way to overcome this problem is to convert the graph structures into a numeric characteristics [7].

Generally speaking, most existing methods can be categorized into two classes, namely 1) embedding graphs into an uncorrelated vectorial space and 2) defining a

graph kernel to characterize the similarity of different graph structures in a high dimensional Hilbert space. Proponents of the first class of methods tend to represent graphs as permutation invariant features in a vector space, where standard machine learning techniques designed for vector-based data can be directly employed. For instance, Wilson et al. [8] have computed graph invariant from algebraic graph theory. They use the spectral decomposition of a Hermitian property matrix as the complex analogue of the Laplacian, and construct symmetric polynomials associated with the eigenvectors. Bunke et al. [7] have embedded graphs into vectors by employing both vertex and edge attributed statistics. Ren et al. [9] have computed vectorial graph features via the Ihara zeta function. They first transform each graph into a directed line graph and then identify the cycles residing on the line graph. The resulting graph features are computed by counting the number of cycles having different lengths. Kondor and Borgwardt [10] have computed invariant graph skew spectrum features, by mapping each graph adjacency matrix into a symmetric group function and computing the associated bispectral invariants. Bai and Hancock [11] have developed a framework to compute depth-based representations of graphs. Specifically, they first decompose each graph into a family of expansion subgraphs with increasing size, and then measure the entropy-based information content of these substructures. Unfortunately, these state-of-the-art methods tend to approximate graph structures in a low dimensional vector space, and thus discard correlation information. By contrast, the proponents of graph kernels can characterize graphs in a high dimensional Hilbert space and thus better preserve the structural correlations of graphs [12].

One of the most generic frameworks of defining graph kernels is the concept of R-convolution proposed by Haussler [13]. The main idea underpinning R-convolution graph kernels is that of decomposing graphs into substructures of limited sizes and then comparing pairs of specific substructures, such as walks, paths, subtrees and subgraphs. Under this scenario, Kashima et al. [14] have defined a marginalized kernel by comparing pairs of random walks associated with vertex and edge labels. Costa and Grave [15] have proposed a pairwise neighborhood subgraph distance kernel, by counting pairs of isomorphic pairwise neighborhood subgraphs. Bach [16] has proposed a point cloud kernel based on a local tree-walk kernel, that is computed by factorizing a

graphical model on the subtrees. Wang and Sahbi [17] have defined a graph kernel for action recognition, by describing actions as directed acyclic graphs (DAGs) and counting the number of isomorphic walk pairs. Harchaoui and Bath [18] have proposed a segmentation graph kernel for images by counting the inexact isomorphic subtree patterns between image segmentation graphs. Bai et al. [19] have defined an aligned subtree kernel by counting pairs of inexact isomorphic subtrees rooted at aligned vertices. Alternative state-of-the-art graph kernels based on R-convolution include a) the subtree-based hypergraph kernel [20], b) the subgraph matching kernel [1], c) the optimal assignment kernel [21], d) the aligned Jensen-Shannon subgraph kernel [22], and e) the fast depth-based subgraph kernel [23].

Unfortunately, as we have stated, the R-convolution graph kernels usually employ substructures of limited sizes. As a result, most of the aforementioned kernels only reflect local graph characteristics, i.e., these kernels cannot capture characteristics of graph structures at the global level. To address this problem, Bai et al. [2, 24] have developed a family of graph kernels based on information theoretic measures, namely the classical and quantum Jensen-Shannon divergence. Specifically, they use either the classical or the quantum random walk associated with the divergence to measure the similarity between graphs at a global level based on entropies. Johansson et al. [25] have developed a global graph kernel based on geometric embeddings. Specifically, they use the Lovász number as well as its associated orthonormal representation to capture the characteristics at a global level.

One common weakness arising in most of the aforementioned state-of-the-art methods, either graph embedding methods or graph kernels, is that of ignoring information from multiple graphs. This is because the graph embedding method usually captures characteristics for each single graph structure. On the other hand, the graph kernel only reflects graph characteristics for each pair of graphs. As a summary, developing effective method to preserve the structural information residing in graphs still remains a challenge.

1.2. Contribution

To overcome the shortcoming of both existing graph embedding methods and graph kernels, in this paper, we aim to present a novel framework for computing deep depth-based representations for graph structures. Our work links the ideas from graph complexity and deep learning networks [4]. In particular, for a set of graphs under analysis, we commence by computing the depth-based representations rooted at each vertex [23]. This is done by decomposing a graph structure into a family of expansion subgraphs of increasing sizes rooted at a vertex, and then measuring the entropy-based complexities of the subgraphs and use this to construct a complexity trace, i.e., the depth-based representation of the root vertex. Since the complexity trace encapsulates entropic information flow from the root vertex to the global graph structure, the depth-based representation can simultaneously capture both local and global graph characteristics.

With the depth-based representations of vertices over all graphs to hand, we employ the well-known k-means method [26] to identify k dominant centroids of these depth-based representation vectors as prototype representations. Since the prototype representations are identified by minimizing the sum of square distances between all depth-based representations and the centroid points (i.e., the prototype representations) of their clusters, the prototype representations can reflect representative characteristics of all depth-based representations encountered. As a result, the prototype representations can encapsulate dominant characteristics over all used graphs.

In order to reflect the high dimensional structural information of graphs well, we propose to further capture the manifolds of these prototype representations using deep learning networks [4]. This is motivated by the recent success of deep learning [3, 27], that has been proven a powerful tool of learning complex relationships of data [28]. Specifically, with the prototype representations as input data, we train a deep neural network (i.e., the deep autoencoder network [3]) that is optimized using Stochastic Gradient Descent together with the Deep Belief Network [29] for pretraining. Note that, training the deep network associated with the smaller set of prototype representations can significantly reduce the burdensome computation associated with depth-based representations of all graphs. Since the deep autoencoder network can minimize the reconstruction error of the output and input prototype representations that encapsulate

the dominant structural information over the vertices of all graphs, the deep network can capture the salient information for these graphs in a highly non-liner latent space. By inputting the depth-based representations of vertices over all graphs to the trained deep network, we compute the deep representation for each vertex. The resulting deep depth-based representation of a graph is computed by averaging the deep representations of all its vertices. We theoretically demonstrate that the deep depth-based representations of graphs not only reflect both the local and global characteristics of graphs through the depth-based representations, but also capture the main relationships and information over all graphs under investigations. Experimental evaluation demonstrate the effectiveness of the proposed method.

1.3. Paper Outline

The remainder of this paper is organized as follows. Section 2 reviews the preliminary concepts that will be used in this work. Section 3 details the proposed framework for computing deep depth-based representations of graphs. Section 4 provides the experimental evaluation of the new method. Section 5 concludes this work.

2. Preliminary Concepts

In this section, we review some preliminary concepts that will be used in this work. We commence by reviewing the concept of depth-based representation from entropy-based complexity measures. Finally, we show how to identify the k dominant centroids of the depth-based representation vectors as prototype representations based on the well-known k-means clustering method.

2.1. Depth-Based Representations

In this subsection, we review how to compute the depth-based representation rooted at each vertex of a graph [23]. The depth-based representation is a powerful tool for characterizing the topological structure in terms of the intrinsic complexity. One way of computing the representation is to gauge the information content flow through a family of K-layer subgraphs of a graph (e.g., subgraphs around a vertex having minimal path length K) of increasing size and to use the flow as a structural signature. By

measuring the entropy measure of each expansion subgraph, Bai et al. [23] have shown how to use this this to characterize each graph as a depth-based complexity trace representation that gauges how the entropy-based complexities of the expansion subgraphs vary as a function of depth.

Specifically, assume a sample graph is denoted as $G(V, E)$, where V and E are the vertex and edge sets respectively. We first compute the shortest path matrix of $G(V, E)$ as S_G based on the well-known Dijkstra's algorithm. Note that S_G follows the same vertex permutation with the adjacency matrix of G and each element $S_G(v, u)$ indicates the shortest path length between vertices u and v. Assume the neighbourhood vertex set of $v \in V$ is denoted as N_v^K and satisfies

$$N_v^K = \{u \in V \mid S_G(v, u) \leq K\}. \tag{1}$$

For each vertex $v \in V$ of $G(V, E)$, a family of K-layer expansion subgraphs $\mathcal{G}_v^K(\mathcal{V}_K; \mathcal{E}_K)$ rooted at v is defined as

$$\begin{cases} \mathcal{V}_v^K &= \{u \in N_v^K\}; \\ \mathcal{E}_v^K &= \{(u, v) \subset N_v^K \times N_v^K \mid (u, v) \in E\}. \end{cases} \tag{2}$$

Note that the K-layer expansion subgraph $\mathcal{G}_v^K$ is the global structure of $G(V, E)$ if K is equal or larger than the length of the greatest shortest path starting from vertex v to the remaining vertices of $G(V, E)$. Fig.1 gives an example to explain how to compute a K-layer expansion subgraph rooted at a sample vertex $\hat{v}_C \in V$. The left-most figure shows the determination of K-layer expansion subgraphs for a graph $G(V, E)$ which hold $|N_{\hat{v}_C}^1| = 6$ and $|N_{\hat{v}_C}^2| = 10$ vertices. While the middle and the right-most figure show the corresponding 1-layer and 2-layer subgraphs regarding the vertex $\hat{v}_C$, and are depicted by red-colored edges. In this example, the vertices of different K-layer subgraphs regarding the vertex $\hat{v}_C$ are calculated by Eq.(1), and pairwise vertices possess the same connection information in the original graph $G(V, E)$.

Definition (Depth-based representations of vertices): For the graph $G(V, E)$, let the family of K-layer expansion subgraphs rooted at each vertex $v \in V$ be denoted as $\{\mathcal{G}_v^1, \cdots, \mathcal{G}_v^K, \cdots, \mathcal{G}_v^L\}$. Based on the definition in [23], the L-dimensional depth-

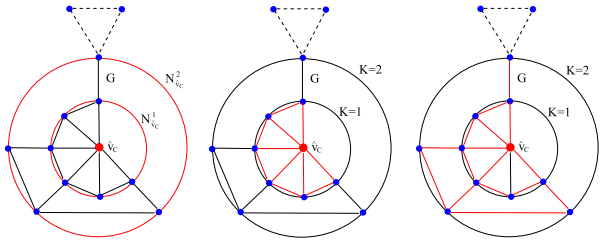

Figure 1: An Example of Computing K-layer Expansion Subgraphs.

based representation DB_v^L of vertex v is defined as

$$\mathrm{DB}_v^L = \{H_S(\mathcal{G}_v^1), \cdots, H_S(\mathcal{G}_v^K), \cdots, H_S(\mathcal{G}_v^L)\}, \tag{3}$$

where $K \leq L$ and $H_S(\mathcal{G}_v^K)$ is the Shannon entropy of $\mathcal{G}_v^K$ computed using the steady state random walk [2]. □

We observe a number of interesting properties for the depth-based representation. First, it is computed by measuring the entropy-based complexity on the gradually expanding subgraphs rooted at a vertex, and thus encapsulates rich intrinsic depth complexity information rooted at the vertex. Second, since the computational complexity of the required Shannon entropy associated with the steady state random walk is only quadratic in the number of vertices [2], the depth-based representation can be efficiently computed [23]. Furthermore, based on Eq.(2), we observe that the family of K-layer expansion subgraphs rooted at a vertex v satisfy

$$v \in \mathcal{G}_v^1 \cdots \subseteq \mathcal{G}_v^K \subseteq \cdots \subseteq \mathcal{G}_v^L \subseteq G.$$

This observation indicates that these expansion subgraphs form a nested sequence, i.e., the sequence of subgraphs gradually expand from the local vertex v to the structure of the global graph. As a result, the depth-based representation can simultaneously capture the local and global graph structure information.

2.2. Identifying Prototype Representations

In this subsection, we identify the centroids over all depth-based representation vectors. In particular, assume a set of N graphs is denoted as $\mathbf{G} = \{G_1, \cdots, G_i, \cdots, G_N\}$. Based on Eq.(3), for each graph G_i we commence by computing the L-dimensional depth-based representations of each vertex of G_i as vertex points. Assume there are

n vertices over all graphs in $\mathbf{G}$, and the L-dimensional depth-based representations of these vertices are $\mathbf{DB}^L = (\mathrm{DB}_1^L, \mathrm{DB}_2^L, \ldots, \mathrm{DB}_n^L)$. We use the k-means method [26] to identify M centroids over all depth-based representations in $\mathbf{DB}^L$, i.e., we divide these representations into M clusters and compute the mean vector for each cluster. Specifically, given M clusters $\Omega = (c_1^L, c_2^L, \ldots, c_M^L)$ where L corresponds to the parameter of these L-dimensional depth-based representations, the k-means method aims to minimize the following objective function

$$\arg\min_{\Omega} \sum_{i=1}^{N} \sum_{\mathrm{DB}_j^L \in c_i^L} \|\mathrm{DB}_j^L - \mu_i^L\|^2, \tag{4}$$

where μ_i^L is the mean of the depth-based representation vectors belonging to cluster c_i^L. Since Eq.(4) minimizes the sum of the square Euclidean distances between the vertex points DB_j^L and their centroid point of cluster c_i^L, the M centroid points $\{\mu_1^L, \cdots, \mu_M^L\}$ can reflect main characteristics of all L-dimensional depth-based representations in $\mathbf{DB}^L$. In other words, these centroid can be seen as a family of L-dimensional prototype representations that encapsulate representative characteristics over all graphs in $\mathbf{G}$.

3. Deep Depth-based Representations of Graphs

In this section, we introduce a framework of computing the deep depth-based representations of graphs, by linking the ideas of depth-based complexity measures and the powerful deep learning networks. We commence by reviewing the concept of deep autoencoder network [29]. Finally, we show how to compute the deep depth-based representation through the deep network.

3.1. Deep Autoencoder Networks

In this subsection, we briefly review the concept of deep autoencoder, that is a deep learning network [29]. This network is one kind of unsupervised model and is composed of a encoder network and a decoder network. The encoder network consists of multiple non-linear functions that map the input data to a representation space, i.e., the encoder network transforms the original data into the deep representations [3]. On

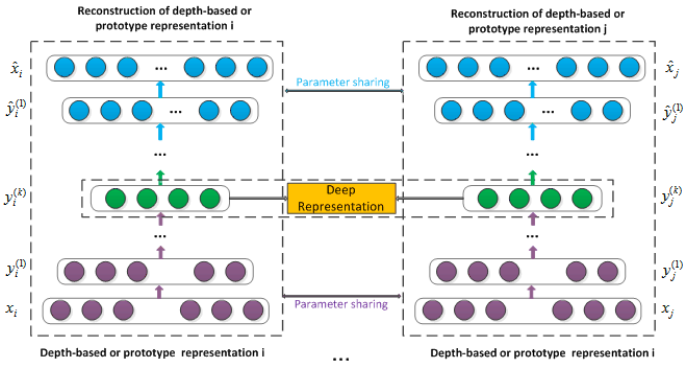

Figure 2: The Architecture of the Deep Autorencode Network.

Table 1: Terms and Notations

Symbol	Definition
n	**number of vertexes**
K	**number of layers**
$S = \{s_1, ..., s_n\}$	**the adjency matrix for the network**
$X = \{x_i\}_{i=1}^n, \hat{X} = \{\hat{x}_i\}_{i=1}^n$	**the input data and reconstructed data**
$Y^k = \{y_i^k\}_{i=1}^n$	**the k-th layer hidden representation**
$W^k, \hat{W}^k$	**the k-th layer weight matrix**
$b^k, \hat{b}^k$	**the k-th layer biases**
$\theta^{(k)} = \{W^k, \hat{W}^k, b^k, \hat{b}^k\}$	**the overall parameters**

the other hand, the decoder network consists of multiple non-linear functions mapping the deep representations in the representation space to the reconstruction space, i.e., the decoder network reconstructs the original data based on the deep representations. It has been proven that the deep representation between the encoder and decoder networks smoothly captures the data manifold embedded in a highly non-liner space. As a result, the deep representation enhances the linear separability of the original data [29], and provides an elegant way of analyzing the original data.

The architecture of the deep autoencouder network used in this work is shown in Fig.2, and the parameters of the deep network are summarized in Table.1. Assume x_i

is the input, then the hidden representations for each layer are computed as

$$y_i^k = \begin{cases} \sigma(W^{(1)}x_i + b^{(1)}) & \text{if } k = 1; \\ \sigma(W^{(k)}y_i^{k-1} + b^{(k)}) & \text{if } k = 2, ..., K. \end{cases} \tag{5}$$

After K steps, we obtain $y_i^{(K)}$ through the encoder network, i.e., we transform x_i into $y_i^{(K)}$ (the deep representation of x_i). Then, after another K steps, we can obtain the output $\hat{x}_i$ by reversing the calculation process of encoder through the decoder network, i.e., we reconstruct the input x_i as $\hat{x}_i$ through $y_i^{(K)}$. The objective of the deep autocoder network is to minimize the reconstruction error of output $\hat{x}_i$ and input x_i, and the loss function is defined as

$$L = \sum_{i=1}^{n} \|\hat{x}_i - x_i\|_2^2 \tag{6}$$

To optimize the deep autocoder network, we first use the Deep Belief Network [29] to pretrain its parameters to avoid trapping in local optimum in the parameter space. Then, the deep network is optimized by means of the Stochastic Gradient Descent method, where the gradients can be conveniently obtained by applying the chain rule to backpropagate error derivatives first through the decoder network and then through the encoder network, i.e., back-propagate $\frac{\partial L}{\partial \theta}$ to update θ^k.

As [3] stated, although the reconstruction process does not explicitly preserve the original input x_i, minimizing the reconstruction error can smoothly capture the manifolds of original data and thus capture the main characteristics of the data.

3.2. The Deep Representation through Deep Networks

In this subsection, we propose a new deep depth-based representation that represents a graph as embedding vector. Let $\mathbf{G} = \{G_1, \cdots, G_i, \cdots, G_N\}$ be a set of graphs. For each graph $G_i(V_i, E_i) \in \mathbf{G}$, we commence by computing the L-dimensional depth-based representation of each vertex $v \in V_i$ based on Eq.(3) as

$$\mathrm{DB}_{i;v}^L = \{H_S(\mathcal{G}_{i;v}^1), \cdots, H_S(\mathcal{G}_{i;v}^K), \cdots, H_S(\mathcal{G}_{i;v}^L)\},$$

where the subscripts i, v and L correspond to the graph $G_i \in \mathbf{G}$, the vertex $v \in V_i$ and the greatest length of the shortest paths over all graphs in $\mathbf{G}$, respectively. Based on the

definition in Section 2.2, we identify a family of L-dimensional prototype representations. Specifically, with the L-dimensional depth-based representations of all graphs in $\mathbf{G}$ to hand, we perform the k-means method to identify M centroids, i.e., we divide the depth-based representations into M clusters and compute the means $\{\mu_1^L, \cdots, \mu_M^L\}$ of these clusters as the L-dimensional prototype representations. We further use the prototype representations $\{\mu_1^L, \cdots, \mu_M^L\}$ as input data to train a deep autoencoder network proposed in Section 3.1. Training the deep network using the smaller set of prototype representations not only avoids the burdensome computation of using depth-based representations of all graphs in $\mathbf{G}$, but also preserves the dominant information of these graphs for the training process. With the trained deep autoencoder network to hand, for each graph $G_i(V_i, E_i) \in \mathbf{G}$, we use its L-dimensional depth-based representations $\mathbf{DBs}_{G_i} = \{\mathrm{DB}_{i;1}^L, \cdots, \mathrm{DB}_{i;v}^L, \cdots, \mathrm{DB}_{i;|V_i|}^L\}$ rooted at all vertices in V_i as input, here the subscripts 1 to $|V_i|$ of $\mathbf{DBs}_{G_i}$ correspond to the 1-th to V_i-th vertex in V_i. Based on Eq.(5) we obtain a set of $|V|$ deep representation vectors of all vertices in V_i as

$$\mathbf{DRs}_{G_i} = \{y_1^{(K)}, \cdots, y_v^{(K)}, \ldots, y_{|V_i|}^{(K)}\}, \tag{7}$$

where $y_v^{(K)}$ is the deep representation of $\mathrm{DB}_{i;v}^L$, i.e., $y_v^{(K)}$ is the deep representation of vertex $v \in V_i$ for G_i. Then, the deep representation of G_i is

$$\mathrm{DR}_{G_i} = \sum_{v \in |V_i|} y_v^{(K)}, \tag{8}$$

i.e., DR_{G_i} is the mean vector of the deep representation vectors in $\mathbf{DRs}_{G_i}$. The resulting deep depth-based representation DDB_{G_i} of each graph $G_i \in \mathbf{G}$ is computed by performing the Principle Component Analysis (PCA) [30] on the graph deep representation matrix $\mathrm{DR}_{\mathbf{G}} = (\mathrm{DR}_{G_1}|\cdots|\mathrm{DR}_{G_i}|\cdots|\mathrm{DR}_{G_N})$ to embed the each deep representation DR_{G_i} of G_i in a principle space. Since the deep representation DR_{G_i} of each graph G_i can effectively capture the manifold of all graphs in the deep space, the deep depth-based representation DDB_{G_i} enhances the linear separability of the graphs. The algorithm of computing the deep depth-based representation is shown in Algorithm 1.

Algorithm 1 Vertex labels strengthening procedure

Input: A set of graphs $\mathbf{G} = \{G_1, \cdots, G_i, \cdots, G_N\}$.

Output: The deep depth-based representation DDB_{G_i} of each graph G_i, and parameters θ of the Deep Autoencoder Network (DAN).

1: Initalization

- For each graph $G_i(V_i, E_i) \in \mathbf{G}$, compute the L-dimensional depth-based representations $\mathrm{DB}^L_{i;v}$ for each vertex $v \in V_i$ based on Eq(3).
- Use the k-means method to divide the depth-based representation vectors of all graphs in $\mathbf{G}$ into M clusters and compute these cluster means $\{\mu^L_1, \cdots, \mu^L_M\}$ as the L-dimensional prototype representations.

2: Train the DAN with Parameters in Table.1.

- Input the prototype representations as the input data X, and pretrain DAN through Deep Belief Network to initialize the parameters $\theta = \{\theta^{(1)}, \cdots, \theta^{(K)}\}$ for DAN.
- **Repeat**
- Based on X and θ, compute $\hat{X}$ using Eq.(5).
- Compute the least square error between $\hat{X}$ and X as $L = \sum_{i=1}^{n} \|\hat{X} - X\|_2^2$.
- Using the Stochastic Gradient Descent to update θ, i.e., back-propagate $\frac{\partial L}{\partial \theta}$ to update θ.
- **Until** converge

3: Compute Deep Depth-based Representations.

- Based on Eq.(7) and (8), compute the deep representation DR_{G_i} for each graph G_i through the trained DAN, perform PCA on the deep representation matrix $\mathrm{DR}_{\mathbf{G}} = (\mathrm{DR}_{G_1}|\cdots|\mathrm{DR}_{G_i}|\cdots|\mathrm{DR}_{G_N})$ to compute the deep depth-based representation DDB_{G_i}.

3.3. Discussions and Related Works

The deep depth-based representation DDB_{G_i} of a graph $G_i \in \mathbf{G}$ proposed in Section 3.2 has a number of advantages.

First, unlike most state-of-the-art graph kernels mentioned in Section 1.1, the proposed deep depth-based representation DDB_{G_i} can simultaneously capture the local and global graph characteristics. This is because the associated depth-based representation for our framework is computed from the entropy measures on the family of expansion subgraphs, that gradually lead a local vertex to the global graph structure. By contrast, the mentioned R-convolution graph kernels [1, 14, 15, 16, 17, 18, 19, 20, 23] are computed by comparing pairs of subgraphs of limited sizes (e.g., paths, cycles, walks, subgraphs and subtrees), and thus only reflect local topological information of graphs. On the other hand, the graph kernels based on the classical and quantum Jensen-Shannon divergence [2, 24], as well as the Lovász graph kernel [25] are computed based on global graph characteristics (e.g., Shannon or von Neumann entropies, and Lovász number associated orthonormal representation). These kernels can reflect global graph characteristics, but tend to ignore local graph information. As a summary, most existing graph kernels cannot reflect complete information of graph structures.

Second, unlike most state-of-the-art graph embedding methods mentioned in Section 1.1, the proposed deep depth-based representation DDB_{G_i} can effectively capture the manifold of the graphs in a highly non-liner latent space, and thus reflect rich characteristics of graph structures. This is because the deep depth-based representation DDB_{G_i} is computed through the powerful deep autoencoder network, that can smoothly capture the data manifold in a highly non-linear space. By contrast, the graph embedding methods [6, 7, 8, 7, 9, 10, 11] tend to embed graphs from high dimensional structure space in a low dimensional vectorial space, and thus lead to information loss.

Third, unlike the state-of-the-art graph kernels and graph embedding methods, the proposed deep depth-based representation DDB_{G_i} can reflect comprehensive characteristics of all graph under investigations. This is because the required deep autoencouder network is trained by using the prototype representations. These representations are identified by employing the k-means clustering method on the depth-based representation vectors of all graphs and computing the centroid of each cluster. S-

ince the prototype representations can represent the main characteristics of all graphs, the deep autoencoder network trained from these representations can simultaneously capture the main information of all graphs. As a result, the deep depth-based representation DDB_{G_i} computed through the deep autoencouder network can encapsulate all graph characteristics. By contrast, most existing graph kernels [1, 14, 15, 16, 17, 18, 19, 20, 23] compute the kernel value based on pairs of graphs. On the other hand, the graph embedding methods [6, 7, 8, 7, 9, 10, 11] compute the graph feature vector based on each individual graph.

Finally, note that, the depth-based complexity traces [11] and the fast depth-based subgraph kernel [23] are also based on the depth-based representations. Thus, similar to the proposed deep depth-based representation, these two existing methods can simultaneously capture the local and global graph characteristics too. However, as one kind of graph embedding method, the depth-based complexity trace can only represent graphs in low dimensional vectorial space, and leads to information loss. By contrast, the proposed method can capture richer graph characteristics through the powerful deep autoencoder network. Furthermore, both the depth-based complexity trace and the fast depth-based graph kernel suffer from the drawback of ignoring comprehensive information over all graphs under investigations, because these methods only capture graph information for each individual graph or pairs of graphs.

The above observations reveal the theoretical effectiveness of the proposed deep depth-based representation. The proposed method provides an effective way of analyzing graph structures for classification or clustering problems.

3.4. Computational Analysis

The computational complexity of the proposed deep depth-based representation is governed by the following computational steps. Consider a set of N graphs each having S vertices and T edges, and the greatest length L of the shortest paths over all these graphs. Computing the depth-based representations of vertices for each graph relies on the calculation of the shortest path matrix, and thus computing the representations of all graphs requires time complexity $O(N\log(S)T)$. The identification of the M prototype presentations relies on performing the k-means method associated with the NS depth-

Table 2: Information of the Graph-based Datasets

Datasets	MUTAG	ENZYMES	BAR31	BSPHERE31	GEOD31	CATH2
Max # vertices	28	126	220	227	380	568
Min # vertices	10	2	41	43	29	143
Mean # vertices	17.93	32.63	95.42	99.83	57.42	308.03
# Graphs	188	600	300	300	300	190
# Classes	2	6	15	15	15	2
# Disjoint graphs	0	31	0	0	0	7
Proportion of disjoint graphs	0%	5.16%	0%	0%	0%	3.68%

Datasets	NCI1	NCI109	COIL5	Shock	PPIs	PTC
Max # vertices	111	111	241	33	218	109
Min # vertices	3	4	72	4	3	2
Mean # vertices	29.87	29.68	144.90	13.16	109.63	25.60
# graphs	4110	4127	360	150	219	344
# classes	2	2	5	10	5	2
# disjoint graphs	580	605	0	0	0	0
Proportion of disjoint graphs	14.11%	14.73%	0%	0%	0%	0%

based representations of all graphs, and thus requires time complexity $O(NSMW)$ where W is the iteration number. Training the deep autoencoder network requires time complexity $O(LDCI)$, where L corresponds to the dimension of input data, D is the degree of the deep network, C is the dimension of the hidden layers and I is the iteration. As a result, computing the deep depth-based representations of all graphs requires time complexity $O(N\log(S)T + NSMW + LDCI)$.

4. Experimental Results

In this section, we empirically evaluate the performance of the proposed deep depth-based representations of graphs. We commence by testing the proposed method on the graph classification problem using standard graph datasets that are abstracted from bioinformatics and computer vision databases. Furthermore, we also compare the proposed method with several state-of-the-art methods, e.g., graph kernels and graph embedding methods.

4.1. Graph Datasets

We demonstrate the classification performance of the proposed method on twelve standard graph-based datasets abstracted from both bioinformatics and computer vision datasets. These datasets include: MUTAG, ENZYMES, BAR31, BSPHERE31, GEOD31, CATH2, NCI1, NCI109, COIL5, Shock, PPIs, and PTC(MR). More details concerning the datasets are shown in Table.2.

MUTAG: The MUTAG dataset consists of graphs representing 188 chemical compounds, and here the goal is to predict whether each compound possesses mutagenicity [31]. The maximum, minimum and average number of vertices are 28, 10 and 17.93 respectively. As the vertices and edges of each compound are labeled with a real number, we transform these graphs into unweighted graphs.

ENZYMES: The ENZYMES dataset consists of graphs representing protein tertiary structures, and contains 600 enzymes from the BRENDA enzyme database [32]. In this case, the task is to correctly assign each enzyme to one of the 6 EC top-level classes. The maximum, minimum and average number of vertices are 126, 2 and 32.63 respectively.

BAR31, BSPHERE31 and GEOD31: The SHREC 3D Shape database consists of 15 classes and 20 individuals per class, that is 300 shapes [33]. This is a standard benchmark in 3D shape recognition. From the SHREC 3D Shape database, we establish three graph datasets named BAR31, BSPHERE31 and GEOD31 datasets through three mapping functions. These functions are a) ERG barycenter: distance from the center of mass/barycenter, b) ERG bsphere: distance from the center of the sphere that circumscribes the object, and c) ERG integral geodesic: the average of the geodesic distances to all other points. Details of the three mapping function can be found in [33]. The number of maximum, minimum and average vertices for the three datasets are a) 220, 41 and 95.42 (for BAR31), b) 227, 43 and 99.83 (for BSPHERE31), and c) 380, 29 and 57.42 (for GEOD31), respectively.

CATH2: The CATH2 dataset has proteins in the same class (i.e., Mixed Alpha-Beta), architecture (i.e., Alpha-Beta Barrel), and topology (i.e., TIM Barrel), but in different homology classes (i.e., Aldolase vs. Glycosidases). The CATH2 dataset is harder to classify, since the proteins in the same topology class are structurally similar. The

protein graphs are 10 times larger in size than chemical compounds, with $200 - 300$ vertices. There is 190 testing graphs in the CATH2 dataset.

NCI1 and NCI109: The NCI1 and NCI109 datasets consist of graphs representing two balanced subsets of datasets of chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines respectively [34]. There are 4110 and 4127 graphs in NCI1 and NCI109 respectively.

COIL5: We establish a COIL5 dataset from the COIL database. The COIL image database consists of images of 100 3D objects. We use the images for the first five objects. For each object we employ 72 images captured from different viewpoints. For each image we first extract corner points using the Harris detector, and then establish Delaunay graphs based on the corner points as vertices. As a result, in the dataset there are 5 classes of graphs, and each class has 72 testing graphs. The number of maximum, minimum and average vertices for the dataset are 241, 72 and 144.90 respectively.

Shock: The Shock dataset consists of graphs from the Shock 2D shape database. Each graph is a skeletal-based representation of the differential structure of the boundary of a 2D shape. There are 150 graphs divided into 10 classes. Each class contains 15 graphs.

PPIs: The PPIs dataset consists of protein-protein interaction networks (PPIs). The graphs describe the interaction relationships between histidine kinase in different species of bacteria. Histidine kinase is a key protein in the development of signal transduction. If two proteins have direct (physical) or indirect (functional) association, they are connected by an edge. There are 219 PPIs in this dataset and they are collected from 5 different kinds of bacteria (i.e., a) *Aquifex*4 and *thermotoga*4 PPIs from *Aquifex aelicus* and *Thermotoga maritima*, b) *Gram-Positive*52 PPIs from *Staphylococcus aureus*, c) *Cyanobacteria*73 PPIs from *Anabaena variabilis*, d) *Proteobacteria*40 PPIs from *Acidovorax avenae*, and e) *Acidobacteria*46 PPIs). Note that, unlike the experiment in [24] that only uses the *Proteobacteria*40 and the *Acidobacteria*46 PPIs as the testing graphs, we use all the PPIs as the testing graphs in this paper. As a result, the experimental results for some kernels are different on the PPIs dataset.

PTC: The PTC (The Predictive Toxicology Challenge) dataset records the carcinogenicity of several hundred chemical compounds for male rats (MR), female rats (FR), male mice (MM) and female mice (FM). These graphs are very small (i.e., $20 - 30$

Table 3: Classification Accuracy (In $\%$ $\pm$ Standard Error) Comparisons

Datasets	MUTAG	ENZYMES	BAR31	BSPHERE31	GEOD31	CATH2
DDBR	**86.57** $\pm$ .33	**42.16** $\pm$ .37	**63.00** $\pm$ .19	**55.00** $\pm$ .20	**42.76** $\pm$ .21	**85.79** $\pm$ .33
DBCT	85.10 $\pm$ .34	38.00 $\pm$ .37	56.00 $\pm$ .20	47.00 $\pm$ .22	36.67 $\pm$ .23	78.42 $\pm$ .41
ISK	84.66 $\pm$ .56	41.80 $\pm$ .43	62.80 $\pm$ .47	52.50 $\pm$ .47	39.76 $\pm$ .43	67.55 $\pm$ .67
JTQK	83.22 $\pm$ .87	39.38 $\pm$ .76	60.56 $\pm$ .35	46.93 $\pm$ .61	40.10 $\pm$ .46	68.70 $\pm$ .69
UQJS	82.72 $\pm$ .44	36.58 $\pm$ .46	30.80 $\pm$ .61	24.80 $\pm$ .61	23.73 $\pm$ .66	71.11 $\pm$ .88
BRWK	77.50 $\pm$ .75	20.56 $\pm$ .35	$--$	$--$	$--$	$--$
WL	82.05 $\pm$ .57	38.41 $\pm$ .45	58.53 $\pm$ .53	42.10 $\pm$ .68	38.20 $\pm$ .68	67.36 $\pm$ .63
SPGK	83.38 $\pm$ .81	28.55 $\pm$ .42	55.73 $\pm$ .44	48.20 $\pm$ .76	38.40 $\pm$ .65	81.89 $\pm$ .63
GCGK	82.04 $\pm$ .39	24.87 $\pm$ .22	22.96 $\pm$ .65	17.10 $\pm$ .60	15.30 $\pm$ .68	73.68 $\pm$ 1.09

Datasets	NCI1	NI109	COIL5	Shock	PPIs	PTC
DDBR	72.58 $\pm$ .45	73.00 $\pm$ .43	**70.27** $\pm$ .29	40.00 $\pm$ .27	76.76 $\pm$ .41	**63.08** $\pm$ .50
DBCT	68.32 $\pm$ .25	68.96 $\pm$ .27	68.33 $\pm$ .30	**44.00** $\pm$ .26	73.26 $\pm$ .43	56.10 $\pm$ .51
ISK	76.21 $\pm$ .25	76.42 $\pm$ .24	38.30 $\pm$ .56	39.86 $\pm$ .68	79.47 $\pm$ .32	60.26 $\pm$ .42
JTQK	**81.23** $\pm$ .25	**81.40** $\pm$ .26	30.86 $\pm$ .66	37.73 $\pm$ .72	**88.47** $\pm$ .47	57.47 $\pm$ .41
UQJS	69.09 $\pm$ .20	70.17 $\pm$ .23	70.11 $\pm$ .61	40.60 $\pm$ .92	65.61 $\pm$ .77	56.70 $\pm$ .49
BRWK	60.34 $\pm$ .17	59.89 $\pm$ .15	14.63 $\pm$ .21	0.33 $\pm$ .37	$--$	53.97 $\pm$ .31
WL	80.68 $\pm$ .27	80.72 $\pm$ .29	33.16 $\pm$ 1.01	36.40 $\pm$ 1.00	88.09 $\pm$ .41	56.85 $\pm$ .52
SPGK	74.21 $\pm$ .30	73.89 $\pm$ .28	69.66 $\pm$ .52	37.88 $\pm$ .93	59.04 $\pm$ .44	55.52 $\pm$ .46
GCGK	63.72 $\pm$ .12	62.33 $\pm$ .13	66.41 $\pm$ .63	26.93 $\pm$ .63	46.61 $\pm$ .47	55.41 $\pm$ .59

vertices), and sparse (i.e., $25 - 40$ edges. We select the graphs of male rats (MR) for evaluation. There are 344 test graphs in the MR class.

4.2. Experiments of Graph Classifications

Experimental Setup: We evaluate the performance of the proposed deep depth-based representations (DDBR) on several standard graph datasets, and then compare them with several alternative state-of-the-art graph kernels and a graph embedding method. The graph kernels used for comparison include: 1) the fast depth-based subgraph kernel based on entropic isomorphism test (ISK) [23], 2) the backtraceless random walk kernel using the Ihara zeta function based cycles (BRWK) [35], 2) the Weisfeiler-Lehman subtree kernel (WL) [36], 3) the shortest path graph kernel (SPGK) [37], 4) the graphlet count graph kernels with graphlet of size 3 (GCGK) [38], 5) the unaligned quantum Jensen-Shannon kernel (UQJS) [24], and 6) the attributed graph kernel from

the Jensen-Tsallis q-differences associated with $q = 2$ (JTQK) [39]. The graph embedding method for comparison is the depth-based complexity trace of graphs (DBCT) [11], and for each dataset the trace dimension number of the DBCT method corresponds to the greatest shortest path length rooted at a vertex to the remaining vertices over all graphs in the dataset, following the concept of the depth-based representations developed in [11].

For the proposed DDBR method, we train a multi-layer deep autoencoder network for each graph dataset, and the dimension of each layer is 500, 250, 100 and 30, i.e., the associated encoder and decoder network both have 4 layer learning structures. Moreover, we set 5% of the vertex number of all graphs in the dataset as the prototype representation number M, e.g., there are 100 vertices of all graphs in a dataset and M is 5. Each deep network is pretained using the Deep Belief Network and then optimized using the Stochastic Gradient Descent. With the trained deep autoencoder network for each dataset to hand, we compute the deep depth-based representation vector as the feature vector for each testing graph. Furthermore, we also compute the depth-based complexity trace as the feature vector for each graph using the DBCT method. We then perform 10-fold cross-validation using the Support Vector Machine Classification (SVM) associated with the Sequential Minimal Optimization (SMO) [40] and the Pearson VII universal kernel (PUK) [41] to evaluate the performance of the proposed DDBR method and the DBCT method. We use nine folds for training and one fold for testing. For each method, we repeat the experiments 10 times. All parameters of the SMO-SVMs were optimized for each method on different datasets. We report the average classification accuracies and standard errors of each method in Table.3. For the WL and JTQK kernels, we set the largest iteration of the required vertex label strengthening methods (i.e., the required tree-index method for the two kernels) as 10. With each kernel to hand, we calculate the kernel matrix on each dataset. We perform 10-fold cross-validation using the C-Support Vector Machine (C-SVM) Classification, and compute the classification accuracies. Similar to the SMO Classification, we also use nine samples for training and one for testing, and each classification was performed with its parameters optimized on each dataset. We report the average classification accuracies and standard errors of each graph kernel in Table.3. Moreover, we also exhibit

the runtime of each graph kernel and graph embedding method on different datasets, and this is measured under Matlab R2011a running on a 2.5GHz Intel 2-Core processor (i.e., i5-3210m). We these results in Table.4. Finally, note that, the JTQK, WL kernels are able to accommodate attributed graphs. In our experiments, we use the vertex degree (not the original vertex labels) as the vertex label for these kernels.

Experimental Results: Overall, in terms of the classification accuracies exhibited by Table.3, the classifications associated with the proposed DDBR method exhibit better performance than state-of-the-art methods for comparisons. Specifically, among the 12 testing graph datasets, the DDBR method achieves the best classification accuracies on 8 datasets, i.e., the MUTAG, ENZYMES, BAR31, BSPHERE31, GEOD31, CATH2, COIL5 and PTC datasets. On the other hand, for the NCI1, NCI109 and PPIs datasets, only the accuracies of the JTQK and WL kernels is obviously better than the proposed DDBR method, and our DDBR method outperforms the remaining methods. Moreover, although the classification associated with the proposed DDBR method does not achieve the best accuracy on the Shock dataset method, the DDBR method is still competitive to the DBCT graph embedding method and the UQJS kernel, and outperforms the remaining methods.

In terms of the runtime exhibited by Table.4, the proposed DDBR method is not the fastest method, but it can still be computed in a polynomial time. By contrast, some graph kernels obviously have more computational time, and even cannot finish the computation in one day. Furthermore, considering the impressive classification accuracies of the proposed DDBR method on most datasets, the proposed method has a good trade off between the classification performance and the computational efficiency.

Experimental Analysis: Table.3 indicates that the proposed method has better performance on classification problems. The reasons for the effectiveness of the proposed DDBR method are threefold. First, the proposed DDBR method can simultaneously capture the local and global graph characteristics, through the required depth-based representations that can lead a local vertex to the global graph structure in terms of entropy measures. By contrast, the alternative graph kernels only capture local or global graph characteristics, and thus lead to information loss. Second, the required deep autoencouder network for the proposed DDBR method is trained by using the prototype

representations that encapsulate the dominant structural information over the vertices of all graphs. Thus, only the proposed DDBR method can reflect comprehensive information of all graph under investigations. By contrast, the alternative DBCT method and the alternative graph kernels can only reflect information of each individual graph or each pair of graphs. Third, the required deep autoencouder network for the proposed DDBR method can minimize the reconstruction error of the output and input prototype representations and effectively capture the manifold of the graphs in a highly non-liner latent space. As a result, the proposed DDBR method based on the deep network can capture the salient information for these graphs in a highly non-liner latent space. By contrast, the alternative DBCT method is one kind of graph embedding method that represents a graph structure in a low dimensional space. Moreover, although the graph kernels can well represent graph structure information in a high dimensional Hilbert space, the proposed DDBR method can smoothly captures the characteristics of graphs through the powerful deep autoencoder network and has better representative power to preserve the graph structure information.

On the other hand, in terms of the less effectiveness of the proposed DDBR method on the NCI1 and NCI109 datasets, Table.2 indicates that there are 14.11% and 14.73% of graphs in the two datasets are disjoint graphs, i.e., some vertices have no path to all the remaining vertices. Since the required depth-based representations for the proposed DDBR method is computed by measuring the entropies on a family of expansion subgraphs rooted at each vertex, the disjoint graph cannot guarantee that its gradually expending subgraphs can accommodate any vertex. In other words, the depth-based representations of disjoint graphs cannot fully capture the whole information of global graph structures. This in turn leads to information loss and influences the effectiveness of the proposed method on the NCI1 and NCO109 datasets. However, even under such a disadvantageous situation, the propose methods still outperform most alternative methods except the JTQK and WL kernels on the two datasets. Furthermore, we observe that only the JTQK and WL kernel can significantly outperform the proposed kernel on the PPIs dataset, since only the two alternative kernels can accommodate vertex labels. The proposed DDBR method outperforms the other alternative methods on the PPIs dataset.

Table 4: CPU Runtime Comparisons

Datasets	MUTAG	ENZYMES	BAR31	BSPHERE31	GEOD31	CATH2
DDBR	68”	$4'50$”	$9'39$”	$8'10$”	$7'35$”	$15'30$”
DBCT	1”	1”	3”	3”	3”	4”
ISK	15”	$3'30$”	$3'50$”	$3'10$”	$2'40$”	$9'51$”
JTQK	3”	30”	$1'22$”	$1'35$”	$1'17$”	$39'14$”
UQJS	20”	$4'23$”	$10'30$”	$13'48$”	$8'49$”	$1h14'$
BRWK	1”	13”	$--$	$--$	$--$	$> 1day$
WL	3”	21”	30”	25”	15”	53”
SPGK	1”	2”	11”	14”	11”	$4'13$”
GCGK	1”	2”	2”	2”	2”	8”

Datasets	NCI1	NCI109	COIL5	Shock	PPIs	PTC
DDBR	$20'30$”	$20'20$”	$10'21$”	45”	$2'10$”	$2'42$”
DBCT	4”	4”	4”	1”	1”	1”
ISK	$2h19'$	$2h20$”	$9'55$”	6”	$1'40$”	59”
JTQK	$10'50$”	$10'55$”	$7'19$”	3”	$1'43$”	8”
UQJS	$2h55'$	$2h55'$	$18'20$”	14”	$3'24$”	$1'46$”
BRWK	$6'49$”	$6'49$”	$16'46$”	8”	$> 1day$	29”
WL	$2'31$”	$2'37$”	$1'5$”	3”	20”	9”
SPGK	16”	16”	31”	1”	22”	1”
GCGK	5”	5”	4”	1”	4”	1”

5. Conclusion

In this work, we have proposed a new framework of computing the deep depth-based representation for graphs. This work is based on the ideas of depth-based graph complexity measures and the powerful deep learning networks. We have identified a family of prototype representations that represent the main characteristics of the deprh-based representations of all graphs. Furthermore, with the prototype representations as input data, we have trained a deep autoencoder network that can capture the main characteristics of all graphs in a highly non-linear deep space. This deep network was optimized using the Stochastic Gradient Descent together with the Deep Belief Network for pretraining. The resulting deep depth-based representation of a graph is computed through the trained deep network associated with its depth-based representations as input. The deep depth-based representations of graphs not only reflect both the local and global characteristics of graphs through the depth-based representations, but also capture the main relationship and information over all graphs under investigation. Experimental evaluations demonstrate the effectiveness of the new proposed method.

Our future work is to extend the proposed method to a new deep graph representation learning method that can accommodate both vertex and edge attributed graphs. Moreover, we will also develop a new framework of computing deep representations for hypergraphs [42] that can preserve higher order information than graphs.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant no. 61503422 and 61602535), the Open Project Program of the National Laboratory of Pattern Recognition (NLPR), and the program for innovation research in Central University of Finance and Economics.

References

[1] N. Kriege, P. Mutzel, Subgraph matching kernels for attributed graphs, in: Proceedings of ICML, 2012.

[2] L. Bai, E. R. Hancock, Graph kernels from the jensen-shannon divergence, Journal of Mathematical Imaging and Vision 47 (1-2) (2013) 60–69.

[3] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of KDD, 2016, pp. 1225–1234.

[4] G. E. Hinton, Deep belief networks, Scholarpedia 4 (5) (2009) 5947.

[5] L. Bai, E. R. Hancock, Fast depth-based subgraph kernels for unattributed graphs, Pattern Recognition 50 (2016) 233–245.

[6] K. Riesen, H. Bunke, Graph classification by means of lipschitz embedding, IEEE Trans. Systems, Man, and Cybernetics, Part B 39 (6) (2009) 1472–1483.

[7] J. Gibert, E. Valveny, H. Bunke, Graph embedding in vector spaces by node attribute statistics, Pattern Recognition 45 (9) (2012) 3072–3083.

[8] R. C. Wilson, E. R. Hancock, B. Luo, Pattern vectors from algebraic graph theory, IEEE Trans. Pattern Anal. Mach. Intell. 27 (7) (2005) 1112–1124.

[9] P. Ren, R. C. Wilson, E. R. Hancock, Graph characterization via ihara coefficients, IEEE Transactions on Neural Networks 22 (2) (2011) 233–245.

[10] R. Kondor, K. M. Borgwardt, The skew spectrum of graphs, in: Proceedings of ICML, 2008, pp. 496–503.

[11] L. Bai, E. R. Hancock, Depth-based complexity traces of graphs, Pattern Recognition 47 (3) (2014) 1172–1186.

[12] M. Neuhaus, H. Bunke, Bridging the Gap between Graph Edit Distance and Kernel Machines, Vol. 68 of Series in Machine Perception and Artificial Intelligence, WorldScientific, 2007.

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

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

[15] F. Costa, K. D. Grave, Fast neighborhood subgraph pairwise distance kernel, in: Proceedings of ICML, 2010, pp. 255–262.

[16] F. R. Bach, Graph kernels between point clouds, in: Proceedings of ICML, 2008, pp. 25–32.

[17] L. Wang, H. Sahbi, Directed acyclic graph kernels for action recognition, in: Proceedings of ICCV, 2013, pp. 3168–3175.

[18] Z. Harchaoui, F. Bach, Image classification with segmentation graph kernels, in: Proceedings of CVPR, 2007.

[19] L. Bai, L. Rossi, Z. Zhang, E. R. Hancock, An aligned subtree kernel for weighted graphs, in: Proceedings of ICML, 2015, pp. 30–39.

[20] L. Bai, P. Ren, E. R. Hancock, A hypergraph kernel from isomorphism tests, in: Proceedings of ICPR, 2014, pp. 3880–3885.

[21] N. M. Kriege, P. Giscard, R. C. Wilson, On valid optimal assignment kernels and applications to graph classification, in: Proceedings of NIPS, 2016, pp. 1615–1623.

[22] L. Bai, Z. Zhang, C. Wang, X. Bai, E. R. Hancock, A graph kernel based on the jensen-shannon representation alignment, in: Proceedings of IJCAI, 2015, pp. 3322–3328.

[23] L. Bai, E. R. Hancock, Fast depth-based subgraph kernels for unattributed graphs, Pattern Recognition 50 (2016) 233–245.

[24] L. Bai, L. Rossi, A. Torsello, E. R. Hancock, A quantum jensen-shannon graph kernel for unattributed graphs, Pattern Recognition 48 (2) (2015) 344–355.

[25] F. D. Johansson, V. Jethava, D. P. Dubhashi, C. Bhattacharyya, Global graph kernels using geometric embeddings, in: Proceedings of ICML, 2014, pp. 694–702.

[26] I. Witten, E. Frank, M. Hall, Data mining: Practical machine learning tools and techniques.

[27] A. Jain, A. R. Zamir, S. Savarese, A. Saxena, Structural-rnn: Deep learning on spatio-temporal graphs, in: Proceedings of CVPR, 2016, pp. 5308–5317.

[28] Y. Bengio, Learning deep architectures for AI, Foundations and Trends in Machine Learning 2 (1) (2009) 1–127.

[29] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, science 313 (5786) (2006) 504–507.

[30] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[31] A. Debnath, R. L. de Compadre, G. Debnath, A. Shusterman, C. Hansch, Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds, correlation with molecular orbital energies and hydrophobicity, J. of Med. Chem. 34 (1991) 786–979.

[32] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, D. Schomburg, The enzyme database: Updates and major new developments, Nucleic Acids Research 32 (2004) 431–433.

[33] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, 3d shape matching through topological structures, in: Proceedings of DGCI, 2003, pp. 194–203.

[34] P. Dobson, A. Doig, Distinguishing enzyme structures from non-enzymes without alignments, J. Mol. Biol. 330 (2003) 771–783.

[35] F. Aziz, R. C. Wilson, E. R. Hancock, Backtrackless walks on a graph, IEEE Transactions on Neural Networks and Learning Systems 24 (6) (2013) 977–989.

[36] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-lehman graph kernels, Journal of Machine Learning Research 1 (2010) 1–48.

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

[38] N. Shervashidze, S. Vishwanathan, K. M. T. Petri, K. M. Borgwardt, Efficient graphlet kernels for large graph comparison, Journal of Machine Learning Research 5 (2009) 488–495.

[39] L. Bai, L. Rossi, H. Bunke, E. R. Hancock, Attributed graph kernels using the jensen-tsallis q-differences, in: Proceedings of ECML-PKDD, 2014, pp. 99–114.

[40] J. C. Platt, Fast training of support vector machines using sequential minimal optimization, Schölkopf, B., Burges, C.J.C., and Smola, A.J. (Eds.) Advances in Kernel Methods (1999) 185–208.

[41] W. Sanders, C. Johnston, S. Bridges, S. Burgess, K. Willeford, Prediction of cell penetrating peptides by support vector machines, PLoS Computational Biology 7 (2011) e1002101.

[42] L. Bai, F. Escolano, E. R. Hancock, Depth-based hypergraph complexity traces from directed line graphs, Pattern Recognition 54 (2016) 229–240.