Applied Network Science

RESEARCH Open Access

# Unsupervised network embeddings with node identity awareness

Leonardo Gutiérrez-Gómez[1*] and Jean-Charles Delvenne[1,2]

*Correspondence:
leonardo.gutierrez@uclouvain.be
[1]Institute for Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM), Université catholique de Louvain, Avenue Georges Lemaitre, 4, 1348 Louvain-la-Neuve, Belgium
Full list of author information is available at the end of the article

**Abstract**

A main challenge in mining network-based data is finding effective ways to represent or encode graph structures so that it can be efficiently exploited by machine learning algorithms. Several methods have focused in network representation at node/edge or substructure level. However, many real life challenges related with time-varying, multilayer, chemical compounds and brain networks involve analysis of a family of graphs instead of single one opening additional challenges in graph comparison and representation. Traditional approaches for learning representations relies on hand-crafted specialized features to extract meaningful information about the graphs, e.g. statistical properties, structural motifs, etc. as well as popular graph distances to quantify dissimilarity between networks.

In this work we provide an unsupervised approach to learn graph embeddings for a collection of graphs defined on the same set of nodes so that it can be used in numerous graph mining tasks. By using an unsupervised neural network approach on input graphs, we aim to capture the underlying distribution of the data in order to discriminate between different class of networks. Our method is assessed empirically on synthetic and real life datasets and evaluated in three different tasks: graph clustering, visualization and classification. Results reveal that our method outperforms well known graph distances and graph-kernels in clustering and classification tasks, being highly efficient in runtime.

**Keywords:** Network embeddings, Graph visualization, Graph classification, Unsupervised learning

## Introduction

Numerous complex systems in social, medical, biological and engineering sciences can be studied under the framework of networks. Network models are often analyzed at the node/edge or substructure level, studying the interaction among entities, identifying groups of nodes behaving similarly or finding global and local connectivity patterns among a given network. Furthermore, many real life challenges might involve collections of networks representing instances of the system under study, e.g functional brain networks (connectomes) (Hagmann et al. 2008), chemical compound graphs (Srinivasan et al. 1997), multilayer networks (Cardillo et al. 2013), and so on. Other applications involve dynamic interactions between components, introducing an additional complexity in the time evolution of the system. For example, in a social mobile phone network, people are considered as nodes and the phone calls as edges. The dynamics of calls between users will systematically add and remove edges between

them, describing a sequence of static graphs characterizing a dynamic evolution of the system.

With the increasing availability of manually labeled network data, many of these problems have recently raised the attention of the machine learning community. Machine learning applications seek to make predictions or discovering patterns in graph structured data. For example, in chemoinformatics (Debnath et al. 1991), one might need to predict the toxicity or anti-cancer activity of proteins and molecules represented as graphs. In time-varying social networks, one might be interested in detecting unusual events (Peel and Clauset 2015), e.g points in time in which the network connectivity differs abruptly with respect to the evolution of the underlying process. Prediction of subjects having a neural disorder such as Alzheimer or Schizophrenia, based on their connectomes is crucial in neuroscience (Griffa et al. 2015).

The cornerstone of this approach is the feature representation of the input data, e.g finding effective ways to encode graph structures in such a way that it can be used in traditional machine learning models. For example, in order to predict whether a molecule is toxic or not, one might build a feature vector representation of a molecule incorporating information about its atoms, as well as global and local properties of the graph structure itself (Barnett et al. 2016; Gutiérrez-Gómez and Delvenne 2019). By doing so we can train a traditional machine learning model such as support vector machines, random forest, neural network, etc. so it will discriminate unseen toxic and non-toxic chemical compounds.

There exist many manners to extract features and comparing networks. For instance, graph distances (Donnat and Holmes 2018; Livi and Rizzi 2013) such as the Jaccard and Hamming distances compute differences between graphs by counting the number of edit operations to transform a graph into another one, focusing mainly in their local connectivity patterns. Other distances are spectral in nature based on the comparison between the eigenvalues of the reference matrices representing the networks. Another popular class of distance measures are the graph kernels (Shervashidze et al. 2011; Yanardag and Vishwanathan 2015). A kernel can often be seen as the scalar product between implicit high-dimensional feature representations of a network (Schölkopf and Smola 2002). The so-called kernel trick allows to compare networks without ever computing explicitly the coordinates of data points in the high-dimensional feature space, sometimes with a substantial gain in computational time over classical graph-distance approaches. Moreover, some supervised approaches, e.g. requiring class labels for graphs, to learn embeddings of entire graphs have shown excellent performances in NLP tasks, i.e. semantic parsing and natural language generation *graph2seq* (Xu et al. 2018), graph classification with CNN for graphs (Niepert et al. 2016) as well as metric learning approaches to brain network classification (Ma et al. 2018). However, they pose some important limitation in practice given the large volume of annotated data needed to learn such deep neural networks architectures, as well as a limited expressiveness of the learned representations constrained to particular supervised tasks.

Because real life networks are complex structures involving diverse connectivity patterns across domains the expressiveness of the aforementioned methods is constrained when applied in multiple tasks. Therefore, the most relevant hand-crafted features tend to

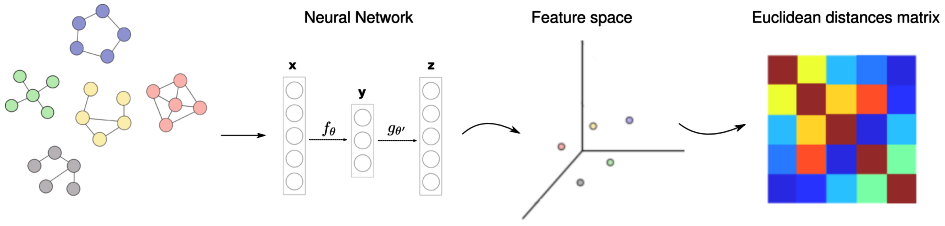


**Fig. 1** Overview of the proposed method. Given a family of graphs, we train an unsupervised neural network in order to uncover dissimilar relationships between graphs. The graphs are embedded into a feature space and mapped to a Euclidean distance matrix reflecting the structural similarity between input examples

be task dependent and often require specialized domain expertise in order to incorporate the right properties to perform accurately on the target task.

Unlike previous approaches, in this work we propose a method to learn unsupervised network embeddings from a collection of networks all defined on the same set of nodes. It should not be confused with node embedding approaches which aim to map nodes from a graph into vectors on a feature space (see Cai et al. (2017); Goyal and Ferrara (2018) for a survey of those methods). Therefore, in this paper we refer to graph or network embedding the outcome of mapping each network of a family as a vector in a Euclidean space (see Fig. 1). The unsupervised nature of the method allows to capture the most relevant features from the data in order to produce lower dimensional representation of input graphs. This reduces the curse of dimensionality of high dimensional graphs uncovering discriminative relationships in the underlying dataset. As a consequence, networks with similar structural properties will have neighboring embeddings in the feature space, and dissimilar graph will be more distant. Our approach thus differs from the various definitions of graph distances or similarities mentioned previously in that we learn automatically a feature representation of graphs assessing their similarity on a Euclidean space, instead of using a hand-crafted metric on the graphs. In addition, because many graph coming from real life applications rarely have exchangeable nodes, we focus on problems defined on networks that account for node identities, e.g. time-varying networks, brain networks, multilayer networks, etc.

Although the purpose of this paper is not to make an extensive evaluation on graph clustering and classification tasks, we rather introduce and illustrate an unsupervised method for embedding networks on same node set, i.e. taking the node identity into account. We validate our method empirically in three network mining tasks: graph clustering (grouping similar graphs together), graph classification (predicting the class to which unseen networks belong to) and visualization (plotting many networks in $\mathbb{R}^2$). We perform diverse experiments on synthetic and real life datasets such as time-varying networks (primary school network), multilayer networks (European airport network) and brain networks datasets.

This paper is structured as follows. First, we introduce some popular methods of the literature used to compare networks, as well as the development of the proposed approach. Then, we present some applications in graph visualization, clustering and classification performed on synthetic and real life datasets. Subsequently, a computational analysis of our method is presented, finalizing with a discussion and perspectives for future work.

## Methods

### Graph distances

Distinguishing among a class of networks requires a notion of distance or similarity between pairs of graphs (Donnat and Holmes 2018). These measures capture different aspects of the local and global structure of graphs having an impact in the outcome of different applications. We present some of the most representative graph distances of the literature.

The *Hamming* and *Jaccard* distances are special instances from the broader class of graph-edit distances. They measure the number edge insertion and deletion operations necessary to transform one graph to another one. Denoting $N$ the number of nodes of the undirected graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with adjacency matrices $A_1$ and $A_2$ respectively, the *Hamming* distance between them is defined as:

$$d_H(G_1, G_2) = \frac{1}{N(N-1)} \sum_{i,j}^{N} |A_1 - A_2|_{i,j} \tag{1}$$

which defines a scaled version of the $L_{1,1}$ norm between matrices bounded between 0 and 1. Similarly, the *Jaccard* distance is defined as:

$$d_J(G_1, G_2) = \frac{|E_1 \cup E_2| - |E_1 \cap E_2|}{|E_1 \cup E_2|} \tag{2}$$

where $E_1$ and $E_2$ are the set of edges for the graphs $G_1$ and $G_2$ respectively.

*DeltaCon* (Koutra et al. 2016) is a popular graph similarity measure in connectomics. As the edit distances it also exploits node correspondence across graphs. The intuition behind the method is to compute first pairwise node similarities of input graphs through a variant of a personalized PageRank algorithm (Koutra et al. 2016). The pairwise node affinity matrices $(S_1, S_2)$ are compared using the Matusita Distance defined by:

$$d_{DC}(S_1, S_2) = \sqrt{\sum_{i,j=1}^{n} \left(\sqrt{S_1(i,j)} - \sqrt{S_2(i,j)}\right)^2} \tag{3}$$

On the other hand, the *spectral distances* for graphs have proven to be very useful in many applications (Masuda and Holme 2019; Wilson and Zhu 2008). However, the spectral nature of the method makes it invariant to node permutations. Roughly speaking, these methods compare the spectrum of any matrix representing the input graph, generally the graph Laplacian. The combinatorial Laplacian matrix (CL) of an undirected graph $G$ is defined by $L = D - A$, where $D$ is the diagonal matrix whose $i$-th element equal to the degree of node $i$, and $A$ its adjacency matrix. The normalized Laplacian matrix (NL) is defined by $L' = D^{-1/2} L D^{1/2} = I - D^{-1/2} A D^{1/2}$, with $I$ the corresponding identity matrix. We denote the eigenvalues of any of the Laplacian matrices as $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N$.

For any $L$ and $L'$ we consider the following spectral distance. The spectral distance between two undirected graphs $G_1$ and $G_2$ is defined as Masuda and Holme (2019):

$$d(G_1, G_2) = \sqrt{\sum_{i=1}^{n_\lambda} [\lambda_{N+1-i}(G_1) - \lambda_{N+1-i}(G_2)]^2} \tag{4}$$

where $n_\lambda$ is the number of eigenvalues considered for the computation of the distance (typically $n_\lambda = N$).

### Embedding distances

Unlike the previous distances, our approach performs network comparisons directly on a feature space through a learned non-linear mapping applied to input graphs (see Fig. 1). The building blocks of our method are explained in the following subsections.

#### *Autoencoder*

Unsupervised learning approaches aim to uncover hidden patterns or learning representations from unlabeled data. The autoencoder (AE) (Vincent et al. 2008) is one of the most popular unsupervised neural network approaches. It has been widely used as a performant mechanism to pre-train neural networks and general purpose feature learning (Choi et al. 2018). It allows to compress the representation of input data, disentangling the main factors of variability, removing redundancies and reducing the dimension of the input.

Given a set of data examples $D = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\}$, the purpose of the the traditional auto-encoder is to learn a non-linear mapping which encodes an input example $\mathbf{x} \in \mathbb{R}^n$ in a smaller dimensional latent vector $\mathbf{y} \in \mathbb{R}^d$ with $n \gg d$. The encoding mapping has the form of $f_\theta(\mathbf{x}) = s(W\mathbf{x} + b) = \mathbf{y}$, generally through a non-linear function $s$ such as sigmoid or *tanh* applied entrywise on the vector $W\mathbf{x} + b$. A reverse mapping of $f$ is used to reconstruct the input from the feature space: $g_{\theta'}(\mathbf{y}) = s(W'\mathbf{y} + b') = \mathbf{z}$. The parameters $\theta = \{W, b\}$ and $\theta' = \{W', b'\}$ are optimized by minimizing the average reconstruction error over the training set:

$$\theta^*, \theta'^* = \underset{\theta,\theta'}{arg\,\min} \ \frac{1}{m} \sum_{i=1}^{m} \left\| \mathbf{x}^{(i)} - \mathbf{z}^{(i)} \right\|_2^2 \tag{5}$$

Note that when $s$ is the trivial identity, the solution is equivalent to the classical PCA (principal component analysis) with the number of hidden units as the principal components. One can therefore see autoencoders as a nonlinear extension of PCA.

#### *Denoising autoencoder (DAE)*

Minimizing the previous reconstruction criterion alone is unable in general to guarantee the extraction of meaningful features as it can potentially memorize the training data. We want the Autoencoder to be sensitive enough to recreate the original observation but insensitive enough to the training data such that the model learns a generalizable encoding and decoding mapping.

To avoid this limitation, the objective (Eq. 5) is redefined in such a way that the autoencoder will be able to clean partially corrupted input or simply denoising it. This modification leads a simple variant of the basic autoencoder described above. A denoising autoencoder (DAE) (Vincent et al. 2008) is trained to reconstruct a clean or repaired version from a corrupted input. This is done by transforming the original input $\mathbf{x}$ in $\tilde{\mathbf{x}}$ through a stochastic mapping $\tilde{\mathbf{x}} \sim q_D(\tilde{\mathbf{x}}|\mathbf{x})$. By doing so the AE is forced to learn meaningful features, robust under corruption of the input.

The corrupted version $\tilde{\mathbf{x}}$ is mapped with the original autoencoder to a hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$ from which we reconstruct a clean $\mathbf{z} = g_{\theta'}(\mathbf{y})$. An important observation is that $\mathbf{z}$ is now a deterministic function of $\tilde{\mathbf{x}}$ rather than $\mathbf{x}$. See Fig. 2 for an schematic representation of the model. Thus, we optimize the same objective than Eq. 5 but replacing $\mathbf{x}$ by $\tilde{\mathbf{x}}$. Optimization is done with the standard mini-batch gradient descent and back propagation algorithms (Lecun et al. 1998).

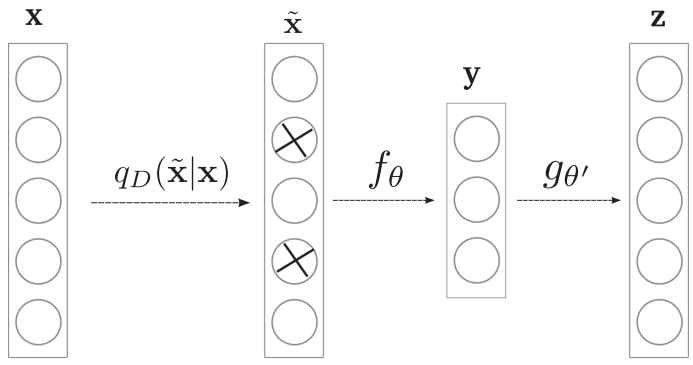


**Fig. 2** Denoising Autoencoder. A corrupted instance $\tilde{x}$ of a graph **x** is fitted into the Autoencoder's input. The Autoencoder is trained to recover a cleaned version of the input by compressing it through a non-linear mapping $f_\theta$ and mapping it back (through $g_{\theta'}$) to a reconstructed version of the original input graph **x**

#### *Network embedding distances*

The adjacency matrix $A$ of a graph is a simple network representation but alone can be insufficient as an input for the DAE. It only captures first order relationships between neighboring nodes. We extend this by computing higher powers of the adjacency matrix in order to capture multiple paths relationships. Thus, we consider $A^r$ for some $r \geq 1$ as a more adequate input for the Denoising Autoencoder.

Note that as the class of problems we tackle are defined on a collection of networks having a node correspondence across graphs, our method remains invariant to the node ordering when the same node permutation is assigned to the graphs.

The vectorization of matrices is required to feed the graphs into the DAE input. Let $A^r$ the $r$ power of the $n \times n$ adjacency matrix $A$ of a graph. The vectorization of $A^r$ is a $n^2 \times 1$ column vector $\mathbf{x} = vec(A^r)$ obtained by staking the columns of $A^r$. Notice that when the graph is undirected, the input matrix can be described with a $\frac{n(n+1)}{2} \times 1$ column vector **x**. We apply a stochastic noise on the input by removing or adding an small fraction of edges at random, then we infer the parameters of the DAE using the noisy inputs $\tilde{\mathbf{x}}$ as was presented in the previous section.

The optimal solution $\theta^* = \{W^*, b^*\}$ parametrizes an encoder mapping $f_{\theta^*}$ of the DAE. It embeds the input $\mathbf{x} = vec(A^r)$ into a smaller dimensional vector $f_{\theta^*}(\mathbf{x}) \in \mathbb{R}^d$. A main advantage of transforming graphs into feature vectors is that it allows us to compare easily networks computing only Euclidean distances between their embeddings. Hence, the *network embedding distance* between two graphs $G_1$ and $G_2$ with power matrices $A_1^r$ and $A_2^r$ is defined as:

$$d(G_1, G_2) = \left\| f_{\theta^*}\left(vec\left(A_1^r\right)\right) - f_{\theta^*}\left(vec\left(A_2^r\right)\right)\right\|_2. \tag{6}$$

In the following we introduce other related work on graph embeddings.

#### *Other graph embeddings*

Inspired by the success of the recent document embedding models (Le and Mikolov 2014), *graph2vec* (Narayanan et al. 2017) extends the same ideas in the graph domain. This method proposes to consider a graph as a document where rooted subgraphs

around every node in the graph are the words composing the document. The underlying assumption is that different subgraphs compose graphs in a similar way that different words compose sentences/documents. The graph embedding is then trained following the popular skip-gram model (Mikolov et al. 2013), sampling rooted subgraphs around different nodes as input vocabulary. These non-linear substructures allow to capture structural equivalence patterns ensuring that *graph2vec* representation learning will yield similar embeddings for structurally similar graphs. Similar to this, *subgraph2vec* (Narayanan et al. 2016) is an unsupervised method to learn subgraph embeddings but sampling linear substructures such as fixed length random walks.

On the other hand Graph Attention Model (*GAM*) (Lee et al. 2018) is an end-to-end approach for graph classification based on Recurrent Neural Networks (RNN). The attention mechanism allows the method to process a portion of the graph avoiding noise in the rest of the graph. The idea is to use a walk on the graph to sample a number of nodes and using an attention mechanism to guide the walk. It helps to biases the walks towards neighboring nodes having similar node labels, exploiting local graph information. At the end, the RNN component will integrate information gathered from different parts of the graph, accumulating the historical representation of the data for predicting the output graph label. In practice, the history layer $h_t$ is taken as graph embedding for our experiments.

It is worth mentioning that we introduce and illustrate an unsupervised method for embedding networks on same node set, i.e. taking the node identity into account. Although we found no exact competitors in the literature, making systematic comparison difficult, it is illuminating to compare it numerically on some typical tasks with edit-distances like *Hamming's* and *Jaccard's* (which are node-id aware but not learning an embedding from a given family of networks), *DeltaCon* which is also node-id aware but not learning an embedding, spectral methods (which are not node-id aware, and not learned), *Graph2Vec* and *Subgraph2vec* (which learn unsupervised embeddings but are not node-id aware) and *GAN*.

In the following sections we present some experimental results of our method in various synthetic and real life applications.

## Experiments and results

The experiments have three purposes. First, they assess the performance of our method in discriminating different types of networks which are generated from different models, edge densities and heterogeneous community structure. Next, they show the use of graph embeddings in networks coming from diverse real life applications such as time-varying networks, connectomes and multilayer networks. Finally, they highlight the runtime performance of feature computation and compare it against other techniques.

It is worth to mention that all our experiments were performed with $A^3$ where $A$ is the adjacency matrix of the graph, as input for the *DAE*. As $A^3$ counts the number of 3−hop walks between pairs of nodes, it will encode a 'mesoscopic' view of the graph beyond immediate neighbor-neighbor relations. For instance it includes on the diagonal the number of triangles, a feature deemed representative of the clustering of the complex networks (Newman 2003).

The dimension of our graph embedding was determined empirically and fixed in all experiments to 800. The dimension for the considered graph embedding methods was

also 800. It was set as the one that provided the best clustering performance. The training hyper-parameters for the *DAE*, *graph2vec*, *GAM* and *subgraph2vec* methods can be found in the appendix section.

We evaluate our approach on three different but related tasks: graph visualization, graph clustering and classification.

#### Synthetic datasets

We generate four synthetic datasets incorporating different structures, degree distributions and edge densities. Each dataset has 600 networks with 81 nodes. An overview of the generated datasets is shown in Table 1. Their properties are detailed in the following:

- In the first synthetic dataset, we generate three Erdős-Rényi networks (*ER*) with different parameters, producing random networks with different average degrees. Then we make 200 copies from each graph reordering the nodes with a different permutation of the original graph.
- In the second dataset named *Mixed*, we generate two groups of networks: a collection of power-law networks generated with the Barabási-Albert (*BA*) model and Erdős-Rényi networks, all of them with the same average degree of 6.
- In the third dataset, power-law networks were generated using the Lancichinetti–Fortunato–Radicchi (*LFR*) benchmark (Fortunato and Lancichinetti 2009). This algorithm creates networks with heterogeneous structures and communities sizes. The mixing parameter $\mu \in [0, 1]$ controls the strength of the community arrangements, achieving well defined communities with small $\mu$, meaningless community structure when $\mu$ is close to one and $\mu = 0.5$ as the border beyond which communities are no longer defined in the strong sense (Radicchi et al. 2004). Thus, we generate two groups of networks: one with mixing parameter $\mu = 0.1$ and other with $\mu = 0.5$. Other parameters are common for both groups: number of nodes $N = 81$, average node degree equal 11, community sizes varying between 6 and 22 nodes, exponent for the degree sequence 2 and exponent for the community size distribution 1. Therefore, the two groups of networks differ only in the strength of their communities structure and not in the degree distribution, being a more challenging problem than the previous dataset.
- In an attempt to simulate a dynamic network evolution, we simulate a time varying network (*Dynamic*) following (Donnat and Holmes 2018), applying a perturbation mechanism from a starting ER network. At each time step, a fraction of edges of the previous graph are rewired uniformly at random. At the same time, we apply a depletion/thickening process in which edges are deleted with probability 0.015 and formerly absent edges are added with probability 0.015. We introduce two perturbation points by augmenting the probabilities of adding and deleting edges to

**Table 1** Summary of synthetic datasets

| DATASET | Type of network | Properties | True clusters |
|---|---|---|---|
| ER | Erdős-Rényi | Different average degrees | 3 |
| Mixed | ER - Power law | Different models, same average degree | 2 |
| LFR | Power law | Strong vs weak communities strength | 2 |
| Dynamic | Erdős-Rényi | Perturbation mechanism: rewiring, adding and removing % edges | 3 |

0.2 from time $t = 200$ and also to 0.6 from time $t = 400$, defining three ground truth clusters of similar behaving networks.

For all datasets we generate balanced ground truth classes.

### Graph visualization

A useful application of network embedding is graph visualization. It mainly consists in representing graph as 2D points, e.g an entire graph as one point, maximizing a certain notion of similarity. Considerable research has been done in visualizing nodes of graphs based on the premise that nodes sharing common structures e.g. neighboring nodes, structural equivalent nodes, assortative nodes, etc. should be mapped to close points in the embedding space (Cai et al. 2017; Goyal and Ferrara 2018).

In contrast, we propose to visualize multiple graphs at once on a two-dimensional space in the following way. From a given family of graphs, their embeddings are learned and used to compute the embedding distance matrix (Eq. 6). The same procedure is applied for the graph embeddings described in Methods. In order to enable a visualization, a methodology is needed to bring the embedding distances into a low-dimensional visualization. We choose the Multi-scale SNE tool (Lee et al. 2015) as standard method. This is a non-linear dimensionality reduction approach for data visualization which aims to reproduce in a low-dimensional space the local and global neighborhood similarities observed on any similarity matrix. In this way, we expect that networks with similar properties as learned by the Denoising Autoencoder are neighboring points in the two dimensional visualization, while the gap between dissimilar groups of graphs is maximized.

We perform a visualization of some datasets from Table 1. As networks within a particular dataset were generated with a range of parameters, we assign a color to each point in the visualization that reflects the value of the parameter used to generate the network. In this way we expect that a good visualization will preserve the same colored points as neighbors points in $\mathbb{R}^2$ maximizing the gap between groups.

#### *Discussion*

Figure 3 shows the visualization of the *ER* dataset after applying our method together with the methods introduced in the aforementioned sections. as can be expected, results

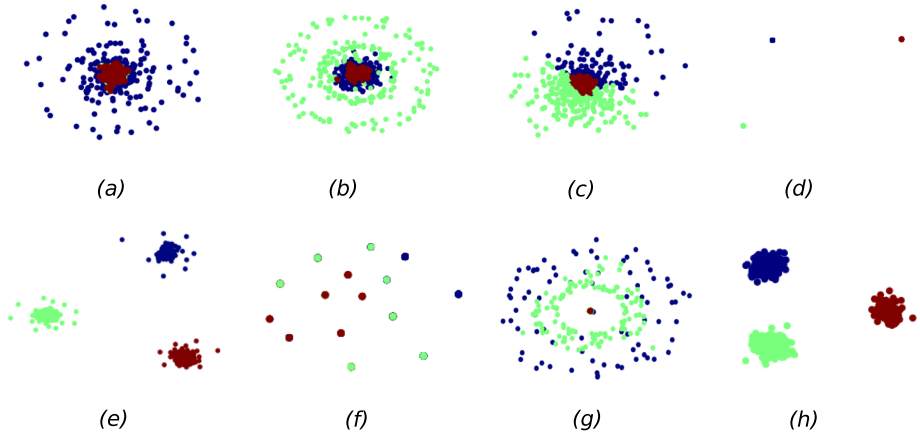


**Fig. 3** Visualization of permuted Erdős-Rényi (ER) dataset. Each point corresponds to a network. Color of a point indicates the category of the network according with its average degree $\langle d \rangle$. For blue $\langle d \rangle = 4$, green $\langle d \rangle = 6$ and red color $\langle d \rangle = 8$ **a** Jaccard. **b** Hamming. **c** Deltacon. **d** Claplacian. **e** Graph2vec. **f** GAM. **g** Subgraph2vec. **h** Emb

with *Jaccard* and *Hamming* distances are not satisfactory because points from different groups overlap. even though *DeltaCon* tries to separate the data, the boundary between groups was not clearly determined. because spectral distances are permutation invariant measures, they collapse all permuted graphs to the same point showing a hard separation between classes. Regarding graph embeddings, *graph2vec* makes a good job grouping the data in three non-overlapping clusters whereas *subgraph2vec* creates concentric cloud of points collapsing different graphs to the center. on the other hand *GAM* captures a different notion of similarity, where graphs with locally similar nodes degree sequences are grouped together. finally, our embedding (*Emb*) shows three well defined clouds of points grouping together isomorphic graphs. our method exploits node correspondence across graphs when it is known, but even if the node order is not of particular significance we can retrieve networks that are essentially identical.

Similarly, in Fig. 4 we show the visualizations of *Mixed* and *LFR* datasets. We can make the following observations: CLaplacian and our graph embedding method (*Emb*) are capable to discover the differences between data examples, so that they separate the data almost perfect clusters. Although edit distances (*Jaccard, Hamming*) work better in discriminating networks with distinct degree distribution than networks with community structure (*LFR*), their discriminative power is still poor. *GAM* as in Fig. 3 collapses networks with local similarities into the same embedding, performing poorly in this benchmark. Even though *subgraph2vec* does not capture perfectly the differences in *ER* and *Mixed* datasets, it performs better than *graph2vec* on networks with community structure.

Finally, Fig. 5 shows the visualization of the *LFR* dataset in more detail. The left-hand side plot shows two clouds of points encoding networks with different mesoscopic structure. As can be seen the blue cluster tends to spread more than the red one, which is more compact. This illustrates the structural variability of networks having heterogeneous number/size communities (blue cluster) against a group of networks with weakly modularity (red cluster).

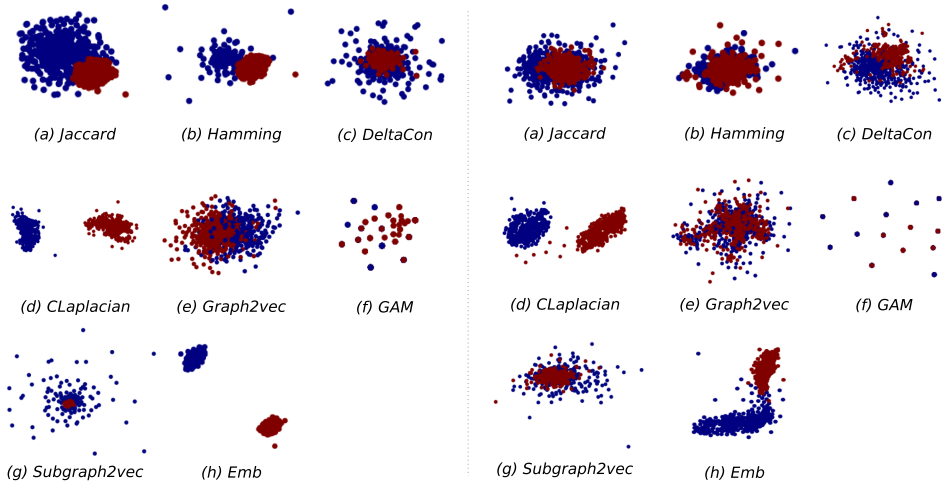


**Fig. 4** *(Left)* networks from *Mixed* dataset. Blue color correspond to *ER* networks and red color are *BA* networks, both groups with $\langle d \rangle = 6$. *(Right)* Networks generated with the LFR benchmark. Blue color correspond to networks with well defined communities, $\mu = 0.1$ whereas red color are networks generated with $\mu = 0.5$

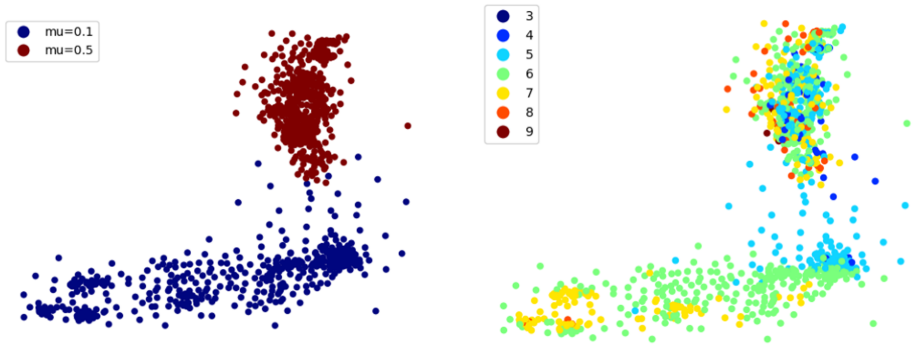


**Fig. 5** Visualization of networks generated with the LFR benchmark. *(Left)* networks with different community strength: $\mu = 0.1$ and $\mu = 0.5$. *(Right)* Same left-hand side networks. Colors encode the number of planted communities within each network

In the right-hand side plot of Fig. 5, we keep the same networks from the left-hand side plot, but we color them according with the number of ground truth communities on each network. Inspecting the bottom cluster we observe that even if there is not a clear grouping of points, the data is distributed in a quasi-continuum manner, having networks with similar number of communities as neighboring points in the plane. On the other hand, the group of networks on the top are indistinguishable, which is expected because their weak community strength. This visualization allows us to understand the notion of similarity captured by the Autoencoder on the underlying dataset.

Once more we emphasize that although the embedding is in principle dependent on the order of the nodes, in this specific case different orderings lead to closely similar visualizations. This is expected as in this case, albeit all the networks are supported on the same number of nodes, there is no natural one-to-one correspondence between the nodes of two networks, and all nodes are treated symmetrically in the generation process.

#### *Visualizing real life networks: temporal networks*

The primary school network (Stehlé et al. 2011) is a dataset containing temporal face-to-face interactions between 232 children and 10 teachers in a primary school in Lyon, France. The data was collected over two days (Thursday, October 1 and Friday, October 2, 2009) spanning from 8:45 am to 5:20 pm the first day, and 8:30 am to 5:05 pm the second day.

The dynamic evolution of the network can be modeled as a time-varying network defined on a fixed number of nodes, and dynamic edges representing the physical interaction between children and teachers. It can be represented as a sequence of static graph snapshots over a time window $\tau$ which aggregates all events or edge activations occurred between the interval $[(t-1)\tau, t\tau]$. For this experiment, we chose a time resolution $\tau = 20s$ yielding 1230 snapshots for Thursday, 01-October. Its visualization is shown in Fig. 6.

The clusters in Fig. 6 can be seen as groups of networks behaving similarly and correlated with external events, e.g consecutive clusters are separated because an external

event. For instance, lunch time is characterized by clusters defined between 12:00 and 14:00. The class time is represented by a long cluster of dark and light blue points in the morning and yellow, orange groups in the afternoon. The end of the school day is highlighted with a brown group of points. Mixed group colors indicates smooth temporal transitions, e.g. end of lunch time and beginning of classes (green-yellow-light blue), also the end of the afternoon break to classes (orange-red).

Note that unlike synthetic examples from the previous section, nodes have an individual identity, and different network snapshots take place on the same set of nodes.

## Graph clustering

### *Clustering synthetic graphs*

Another important application is clustering of networks. Clustering aims to group a set of networks in such a way that networks in the same group (cluster) are more "similar" to each other than to those in other clusters. We proceed similarly to the previous section, but we do not perform dimensionality reduction to $\mathbb{R}^2$. Instead, clustering is performed directly in the embedding space with the standard spectral clustering algorithm (Ng et al. 2002). This technique makes use of the spectrum of a similarity matrix of the data to performing clustering in fewer eigenvectors.

We run our method on each dataset from Table 1 and compute a $600 \times 600$ network embedding matrix using Eq. 6. In order to compare against other techniques, the graph distance matrices for the methods introduced in the first part of the manuscript

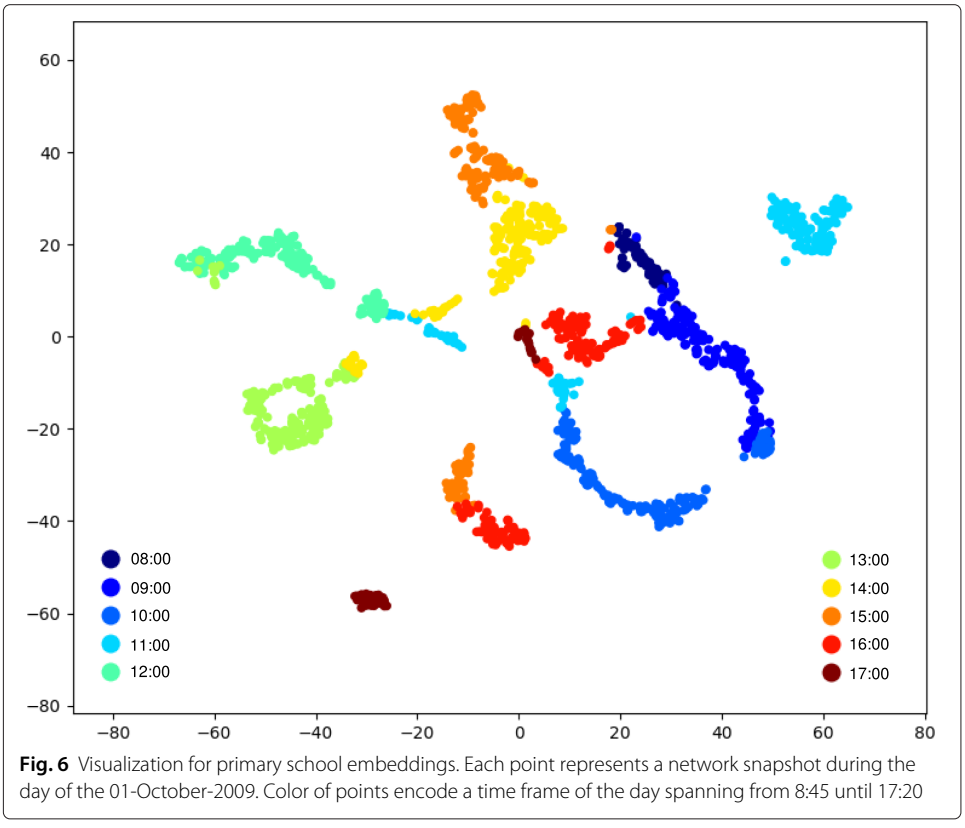


**Fig. 6** Visualization for primary school embeddings. Each point represents a network snapshot during the day of the 01-October-2009. Color of points encode a time frame of the day spanning from 8:45 until 17:20

are computed. All matrices are normalized having a maximum entry of one for similar pairs of graphs and zero for the most dissimilar ones. Therefore, spectral clustering is performed on the similarity matrices induces by graph distances and other graph embeddings.

The clustering performance is evaluated through the normalized mutual information (NMI) (Cahill 2010) metric in the form:

$$NMI(C_t, C_p) = \frac{2I(C_t; C_p)}{H(C_t) + H(C_p)} \tag{7}$$

where $H$ is the entropy of a class distribution and $I$ the mutual information between the ground truth class distribution $C_t$, and the predicted cluster assignment $C_p$. It runs from zero when the algorithm fails to a value of one when the clustering is perfectly recovered. Details about the datasets are presented in Table 1.

In order to evaluate the sensitivity of the clustering to the node ordering, we perform clustering with different enumeration of nodes by applying a fixed node permutation across the networks. We reported the mean and standard deviation of the *NMI* after running the experiment ten times.

#### *Discussion*

Regarding the clustering results in Table 2, we observe that our graph embeddings (*Emb*) provides better clustering than traditional graph distances. The method is capable to differentiate networks with different edge densities (*ER*). Meanwhile, it is also able to discriminate networks with different degree distributions even if they have a similar average degrees (*Mixed*). Discriminating power law networks from strong to weak community structure (*LFR*) is also well achieved. The time-evolving network (Dynamic) is a harder setting in which our method performs the best comparatively to graph distances. In this case the graph embeddings are able to capture the variations introduced by change points in the underlying evolution of the network. This can be explained because the DAE was not designed for a target kind of graphs. Instead, it learns the underlying distribution of the data, identifying the main factor of variability adapting its parameters for discriminating networks with different structure. The quality of the embeddings remains almost the same after permuting the nodes, which is confirmed by the low variance in the NMI. Hence, in practice we fixed a node numbering for the learning procedure.

In addition, as can be seen in Table 3 all graph embedding methods have a good performance on the *ER* dataset except *GAM*. We found that globally *subgraph2vec* performs better than *graph2vec*, but our embedding approach (*Emb*) performs the best.

**Table 2** Graph distances

| | Hamming | Jaccard | DeltaCon | CLP | CLP normed | Emb |
|---|---|---|---|---|---|---|
| ER | 0.239 | 0.603 | 0.581 | **1.0** | **1.0** | **1.0 ±0.0** |
| Mixed | **1.0** | **1.0** | **1.0** | **1.0** | 0.364 | 1.0 ±0 |
| LFR | 0.269 | 0.284 | 0.278 | 0.766 | **0.973** | **0.995 ±0.005** |
| Dynamic | 0.389 | 0.232 | 0.205 | 0.237 | 0.195 | **0.452 ±0.085** |

Clustering results for synthetic datasets (NMI)

**Table 3** Graph embeddings

| | Graph2vec | GAM | Subgraph2vec | Emb |
|---|---|---|---|---|
| ER | **1.0** | 0.11 | **1.0** | **1.0 ±0.0** |
| Mixed | 0.432 | 0.004 | **0.983** | **1.0 ±0** |
| LFR | 0.002 | 0.0001 | 0.009 | **0.995 ±0.005** |
| Dynamic | 0.074 | 0.001 | 0.219 | **0.452 ±0.085** |

Clustering results for synthetic datasets (NMI)

#### *Clustering real life networks: multilayer networks*

The European Air Transportation Network (ATN) (Cardillo et al. 2013) is a multilayer network with 37 layers each representing a different European airline. Each layer has the same number of nodes which represent 450 European airports. We learn graph embeddings for all layers and we cluster them applying a standard hierarchical clustering algorithm on the network embedding distance matrix. The hierarchical clustering provides partition of layers according with their similarity on the embedding space, see Fig. 7.

Our findings confirm those introduced in Cardillo et al. (2013). We can identify two main clusters representing major and low-cost aerial companies, as well as some regional airlines grouped together. Indeed, these airlines have developed according with different structural/commercial constraints. Low-cost companies tends to avoid being centralized and cover more than one country simultaneously. Major airlines have a hub and spoke network, connecting outlying airports to few central ones, providing a maximum coverage from their home country.

### Graph classification

We evaluate graph classification in the context of supervised classification. It requires previously annotated reference samples (graphs) in order to train a classifier and subsequently classify unknown data.

#### *Brain connectomes classification*

In this experiment we apply our method on a brain networks (connectomes) dataset built from magnetic resonance imaging (MRI) (Chiêm and Crevecoeur 2018). Structural and diffusion MRI data of 91 healthy men and 113 healthy women is preprocessed in order to create undirected networks. All graphs have the same 84 nodes representing neural Regions of Interests (ROIs). Weighted edges correspond to the number of neural fibers linking two ROIs. The ROI keeps the same correspondence among graphs. The task is to classify connectomes according to gender, male or female.

#### *Experimental setup*

We assess the performance of our method against some well known algorithms for graph classification, mainly graph kernels and feature-based approaches. We choose the Shorthest Path (SP) and the Weisfeiler-Lehman (WL) subtree kernels (Shervashidze et al. 2011). We also compare against the feature-based (FB) method (Barnett et al. 2016) and Multi-hop assortativities features (MaF) for network classification (Gutiérrez-Gómez and Delvenne 2019). Such methods provide a pairwise similarity matrix between networks in

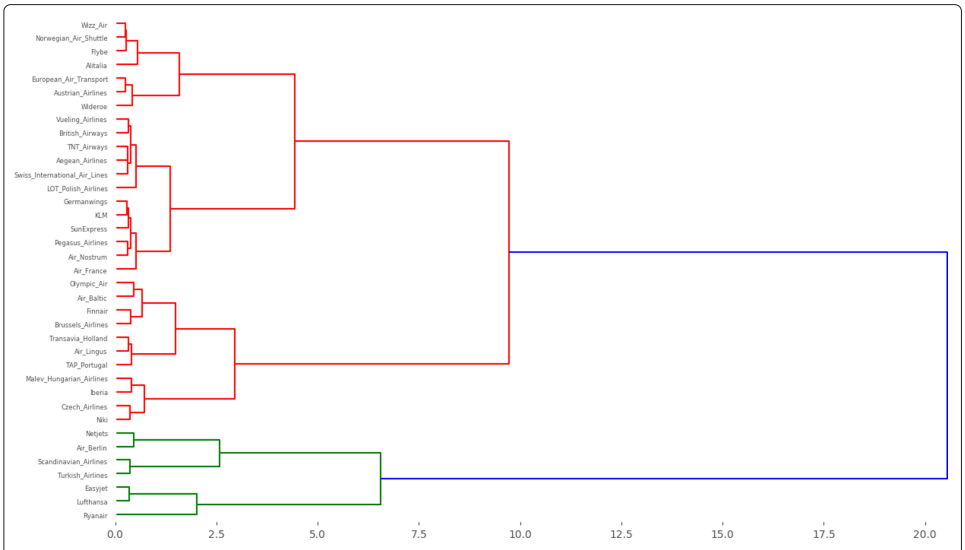


**Fig. 7** Dendrogram of airlines for European airports

the form of a Gram matrix which is used to train a popular support vector machine classifier (SVM) (Boser et al. 1992). Note that the graph distances considered in this work do not define a proper positive semi-definite matrix. Therefore, following (Wu et al. 2005) we shift the spectrum of their similarity matrices providing a proper kernel coherent with the SVM setting.

We follow the experimental setup of (Shervashidze et al. 2011; Yanardag and Vishwanathan 2015). The dataset is randomly split in training and testing sets. The best model is cross-validated over 10 folds. Parameters of SVM are optimized only on the training set. Thus, we compute the generalization accuracy on the unseen test set. In order to exclude the random effect of the data splitting, we repeated the whole experiment 10 times. Finally, we report the average prediction accuracies and its standard deviation.

For each graph kernel we report the result for the parameter that gives the best classification accuracy. For the feature-based approach (Barnett et al. 2016), feature vectors were built with the same network features they reported in their paper: number of nodes, number of edges, average degree, degree assortativity, number of triangles and global clustering coefficient. Results are shown in Tables 4 and 5.

***Discussion***

As can be seen in Table 4, WL, SP and FB perform significantly worse than spectral distances and graph embedding. This is expected as they do not take the identity of the nodes into account. Here, all brains share the same anatomical regions, which make the order of the nodes relevant. In Table 5 can be seen that among the approaches exploiting node correspondence, our method (Emb) outperforms all others while remaining competitive with DeltaCon.

**Table 4** Mean and standard deviation of classification accuracies on brain connectomes dataset

| WL | SP | FB | CLaplacian | NLaplacian | Emb |
|---|---|---|---|---|---|
| 61.20 ± 2.16 | 65.45 ± 1.78 | 65.95 ± 2.54 | 74.19 ± 11.16 | 71.07 ± 10.95 | **87.20** ± **7.60** |

**Table 5** Mean and standard deviation of classification accuracies on brain connectomes dataset

| Hamming | Jaccard | DeltaCon | MaF | Emb |
|---|---|---|---|---|
| 84.37 ± 9.26 | 84.34 ± 10.11 | **87.80 ± 6.54** | 84.26 ± 5.81 | **87.20 7.60** |

*Bold values correspond to the most performing techniques

### Computational cost

Our graph embedding approach involves globally two steps: learning graph embedding through the DAE followed by a pairwise Euclidean distance matrix computation (Eq. 6). For the considered methods listed in Table 2 (except *Emb*), we measure the running time for deriving the pairwise distance matrix. For the graph embedding approaches from Table 3 including our approach (*Emb*), we report the running time for the entire process, including learning the graph embeddings and computing the Euclidean distances. Results are shown below in Fig. 8.

As can be seen our method (*Emb*) outperform all graph distances across all studied datasets. *Hamming* and *Jaccard* distances are globally fast methods because they rely in simple edge-level computations being slower in dense networks. It is known that spectral distances (*CLP, NLP*) are heavy in computation due to the entire eigenvalue calculation. Even if *Deltacon* is a scalable graph similarity measure, it is outperformed by the edit distances (*Hamming, Jaccard*), but prevails over spectral distances. On the other hand, *graph2vec* outperforms the hand-crafted spectral and DeltaCon methods being efficient for a method without node identity awareness. *Subgraph2vec* is globally the slowest method, showing that modeling varying size context for the skip-gram model is considerable more expensive than fixed size neighborhoods. Meanwhile, our graph embedding method remains the fastest. Indeed, the performance of mini-batch gradient descent is

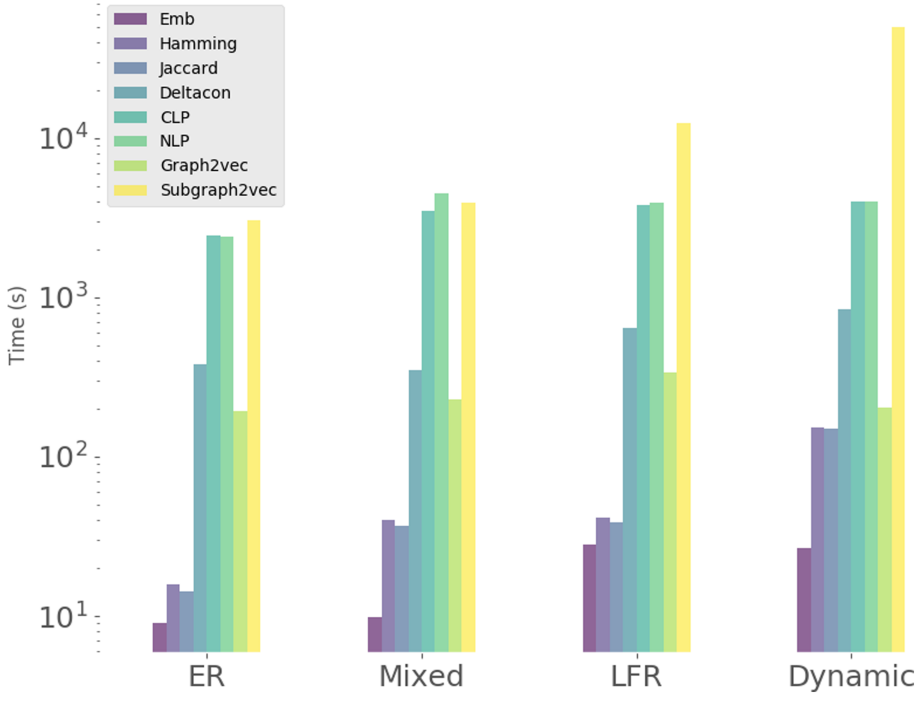


**Fig. 8** Computational time for feature computation. Time is log scaled

proportional to the batch size thus it converges faster on relatively small datasets. For efficiency reasons, the Euclidean distance between two feature embeddings $x, y$ was computed as $d(x,y) = \sqrt{\langle x,x\rangle - 2\langle x,y\rangle + \langle y,y\rangle}$. This formulation has the advantage of being very efficient for sparse graphs given that some terms can be pre-computed for an entire pairwise computation.

All computations were done on a standard computer Intel(R) Core(TM) i7-4790 CPU, 3.60GHzI with 16G of RAM.

## Discussion and concluding remarks

In the presented work we propose a method to learn graph embeddings for a collection of networks defined on the same set of nodes, e.g mapping graphs to $\mathbb{R}^p$ vectors. Our method allows to compare graphs computing only Euclidean distances between their embeddings in a feature space. We evaluate our method in three different applications in graph clustering, visualization and classification. Across diverse synthetic and real life datasets, we compare our approach against well known graph distances, graph-kernel methods and unsupervised graph-embedding methods of the literature.

It turns out that our approach extract the most appropriated features for distinguish different kind of graphs. Indeed, clustering groups of similar networks provides good quality partitions among synthetic datasets (Table 2), discriminating heterogeneous structures among networks better. Although there is not a clear agreement about the use of combinatorial or normalized Laplacian in graph mining applications, spectral distances are highly competitive in graph clustering and visualization but are incapable to exploit node correspondence. Nevertheless, our learned graph embeddings turns out to be computationally cheaper than the considered methods (Fig. 8), being an attractive yet efficient method for comparing networks with equal size.

The results in graph classification reveal that our approach has superior performance than graph-kernels and graph spectral distances (Table 4). Indeed, exploiting the node identities across graphs increases the accuracy of the method. Thus, this result suggest a promising research direction in the connectomics domain.

Note that in this work we were not focusing in the task of for instance differentiating random networks with different average degrees, which can be trivially solved without any machine learning tool. Instead, we aimed to show an automatic way to leave the machine figure out the most relevant hidden patterns from the data, which is more general than designing tailored methods for particular applications.

The current study was limited by the assumption that all networks must have the same set nodes. Even if in many real applications this hypothesis holds, a large amount of complex systems have varying size graphs, e.g. chemical compounds, social networks, etc. This study has only investigated the class of graphs without node/edge attributes, such as age, gender in social networks. In addition, regarding the scalability on large networks, addressing these issues introduce additional challenges and new opportunities for further research.

Despite this limitation, our work has the potential of being extended in two directions. Because the DAE captures the underlying probability distribution of the data (Vincent et al. 2008), the decoding function could be used to generate artificial data, e.g generating brain networks, for mining purposes. Another possibility is to explore deeper

neural network architectures such as the stacked autoencoders (Vincent et al. 2010) and its variants in order to learn hierarchical feature representation of the data for graph classification and clustering applications.

## Appendix

For completeness we provide the parameters we used on our experiments.

| Layer | Number of neurons | Activation function |
|---|---|---|
| Input | $\frac{N(N+1)}{2}$ | – |
| Hidden | 800 | $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ |
| Output | $\frac{N(N+1)}{2}$ | sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$ |

### DAE parameters

- Let $A$ the adjacency matrix of a graph. We consider $A^3$ as input for the DAE.
- Denoting as $N$ the number of nodes of the considered graph, the neural network architecture is:
- We inject 10% of random edges to each example in order to introduce corrupted examples for the denoising process.
- Optimzation: Adam algorithm with initial learning rate of 0.001, a batch of 128 examples and 15 epochs.

### Graph2vec parameters

We used the following parameters: Batch size: 128, epochs: 15, embedding size: 800, Number of negative samples to be used in training: 10, learning rate: 0.3, degree of rooted subgraph (for WL kernel): 3.

### GAM parameters

As this method requires explicitly node attributes, we used the node degree as structural attribute. Optimization was done with the following parameters: Number of training epochs 15, number of neurons in the step network as 32, number of neurons in the shared layer of the step net 64, batch size of 128, time budget for steps 10, learning rate of 0.001 and weight decay $10^{-5}$.

### Subgraph2vec parameters

The parameter used for training this model were: batch size of 128, epochs 3, embedding size 800, learning rate for optimization 1.0, negative sub sampling of 10 and weight of WL kernel 3.

### Synthetic datasets

In the following we provide the parameters used to generate the synthetic datasets. The considered number of nodes is $N = 81$ and 600 examples for each case.

- ER: Erdős-Rényi networks. We generate networks with average degree equal to 4,6,8 and 10. Parameter $p\{0.05, 0.075, 0.10, 0.125\}$
- Mixed: Power law and random networks. We use the Barabási-Albert model with $m = 3$ and Erdős-Rényi model with $p = 0.075$, generating networks with an average degree of six.

- LFR: we generate power law networks with $\mu = 0.1$ and $\mu = 0.5$. The average node degree is 11, community sizes between 6 and 22 nodes, the exponent for the degree distribution equal to 2 and the exponent for community size distribution of 1.
- Dynamic: initial random network with $p = 0.08$. We remove and add edges with a probability of 0.015. At time $t = 200$ we remove and add edges with probability of 0.2, increasing it at time $t = 400$ to probability of 0.6

**Acknowledgements**
We thank Leto Peel and Michel Fanuel for helpful discussion and suggestions. Thanks to Cyril De Bodt for providing the Multi-scale SNE scripts and Benjamin Chiêm for providing the brain dataset. We also thank the anonymous referees for helpful suggestions.

**Authors' contributions**
LG implemented the code, manipulated the data and performed the computational research. JCD contributed with the designing of experiments and performed an examination of results. All authors participated in writing the manuscript. All authors read and approved the final manuscript.

**Funding**
Funding This work was supported by Concerted Research Action (ARC) supported by the Federation Wallonia-Brussels Contract ARC 14/19-060 and Flagship European Research Area Network (FLAG-ERA) Joint Transnational Call "FuturICT 2.0" to which are gratefully acknowledged.

**Availability of data and materials**
The datasets generated and analyzed during the current study are available in the Github repository https://github.com/leoguti85/GraphEmbs The human brain networks dataset used and analyzed during the current study are available from the corresponding author on reasonable request.

**Competing interests**
The authors declare that they have no competing interests.

**Author details**
[1]Institute for Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM), Université catholique de Louvain, Avenue Georges Lemaitre, 4, 1348 Louvain-la-Neuve, Belgium . [2]Center for Operations Research and Econometrics (CORE), Université catholique de Louvain, Avenue Georges Lemaitre, 4, 1348 Louvain-la-Neuve, Belgium.

Received: 1 March 2019 Accepted: 23 August 2019
Published online: 17 October 2019

**References**

Masuda N, Holme P (2019) Detecting sequences of system states in temporal networks. Sci Rep 9(1)
Stehlé J, Voirin N, Barrat A, Cattuto C, Isella L, Pinton J, Quaggiotto M, Van den Broeck W, Régis C, Lina B, Vanhems P (2011) High-resolution measurements of face-to-face contact patterns in a primary school. PLOS ONE 6(8):23176. https://doi.org/10.1371/journal.pone.0023176
Hagmann P, Cammoun L, Gigandet X, Meuli R, Honey CJ, Wedeen VJ, Sporns O (2008) Mapping the structural core of human cerebral cortex. PLOS Biol 6(7):1–15. https://doi.org/10.1371/journal.pbio.0060159
Cahill ND (2010) Normalized measures of mutual information with general definitions of entropy for multimodal image registration. In: Fischer B, Dawant BM, Lorenz C (eds). Biomedical Image Registration. Springer, Berlin, Heidelberg. pp 258–268
Donnat C, Holmes S (2018) Tracking network dynamics: A survey using graph distances. Ann Appl Stat 12(2):971–1012. https://doi.org/10.1214/18-AOAS1176. https://projecteuclid.org/euclid.aoas/1532743483
Schölkopf B, Smola A (2002) Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge. Max-Planck-Gesellschaft
Barnett I, Malik N, Kuijjer ML, Mucha PJ, Onnela J.-P. (2016) Feature-based classification of networks. http://arxiv.org/abs/1610.05868
Chiêm B, Crevecoeur JCDF (2018) Supervised Classification of Structural Brain Networks Reveals Gender Differences. In: 2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)
Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. COLT '92. ACM, New York, NY, USA. pp 144–152. https://doi.org/10.1145/130385.130401. http://doi.acm.org/10.1145/130385.130401
Cardillo A, Gómez-Gardeñes J, Zanin M, Romance M, Papo D, del Pozo F, Boccaletti S (2013) Emergence of network features from multiplexity. Sci Rep 3:1344. http://arxiv.org/abs/1212.2153. https://doi.org/10.1038/srep01344
Ma G, Ahmed NK, Willke TL, Sengupta D, Cole MW, Turk-Browne NB, Yu PS (2018) Similarity learning with higher-order proximity for brain network analysis. CoRR abs/1811.02662. http://arxiv.org/abs/1811.02662
Cai H, Zheng VW, Chang KC (2017) A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. IEEE Trans Knowl Data Eng 30(9):1616–1637. https://doi.org/10.1109/TKDE.2018.2807452

Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Syst 151:78–94. https://doi.org/10.1016/j.knosys.2018.03.022
Choi H, Cho K, Bengio Y (2018) Fine-grained attention mechanism for neural machine translation. Neurocomputing 284:171–176. https://doi.org/10.1016/j.neucom.2018.01.007
Debnath AK, Lopez de Compadre RL, Debnath G, Shusterman AJ, Hansch C (1991) Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. J Med Chem 34(2):786–797. https://doi.org/10.1021/jm00106a046. https://doi.org/10.1021/jm00106a046
Griffa A, Baumann PS, Ferrari C, Do KQ, Conus P, Thiran J.-P., Hagmann P (2015) Characterizing the connectome in schizophrenia with diffusion spectrum imaging. Human Brain Mapping 36(1):354–366. https://doi.org/10.1002/hbm.22633. https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbm.22633
Gutiérrez-Gómez L, Delvenne J.-C. (2019) Multi-hop assortativities for network classification. J Compl Netw 7(4):603–622. https://doi.org/10.1093/comnet/cny034
Newman M (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256. https://doi.org/10.1137/S003614450342480
Fortunato S, Lancichinetti A (2009) Community detection algorithms: A comparative analysis: Invited presentation, extended abstract. In: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools. VALUETOOLS '09. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels. pp 27–1272. http://dl.acm.org/citation.cfm?id=1698822.1698858
Koutra D, Shah N, Vogelstein JT, Gallagher B, Faloutsos C (2016) Deltacon: Principled massive-graph similarity function with attribution. ACM Trans Knowl Discov Data 10(3):28–12843. https://doi.org/10.1145/2824443
Livi L, Rizzi A (2013) The graph matching problem. Pattern Anal Appl 16(3):253–283. https://doi.org/10.1007/s10044-012-0284-8
Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. pp 2278–2324
Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. ICML'14. JMLR.org. pp 1188–1196. http://dl.acm.org/citation.cfm?id=3044805.3045025
Lee JB, Rossi R, Kong X (2018) Graph classification using structural attention. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '18. ACM, New York. pp 1666–1674. https://doi.org/10.1145/3219819.3219980. http://doi.acm.org/10.1145/3219819.3219980
Lee JA, Peluffo-Ordóñez DH, Verleysen M (2015) Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure. Neurocomputing 169(Complete):246–261. https://doi.org/10.1016/j.neucom.2014.12.095
Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13. Curran Associates Inc, USA. pp 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959
Narayanan A, Chandramohan M, Chen L, Liu Y, Saminathan S (2016) subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. In: MLGWorkshop. KDD'16 Workshop
Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y (2017) graph2vec: Learning distributed representations of graphs. In: 15th International Workshop on Mining and Learning with Graphs. MLGWorkshop 2017
Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: Proceedings of The 33rd International Conference on Machine Learning, vol. 48. PMLR. pp 2014–2023. http://arxiv.org/abs/1605.05273
Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: Analysis and an algorithm. In: Dietterich TG, Becker S, Ghahramani Z (eds). Advances in Neural Information Processing Systems 14. MIT Press. pp 849–856. http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm.pdf
Peel L, Clauset A (2015) Detecting change points in the large-scale structure of evolving networks. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI'15. AAAI Press, Austin. pp 2914–2920. http://dl.acm.org/citation.cfm?id=2888116.2888122
Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D (2004) Defining and identifying communities in networks. Proc Nat Acad Sci 101(9):2658–2663. https://doi.org/10.1073/pnas.0400054101. https://www.pnas.org/content/101/9/2658.full.pdf
Shervashidze N, Schweitzer P, van Leeuwen EJ, Mehlhorn K, Borgwardt KM (2011) Weisfeiler-lehman graph kernels. J Mach Learn Res 12:2539–2561
Srinivasan A, King RD, Muggleton SH, Sternberg MJE (1997) The predictive toxicology evaluation challenge. In: Proceedings of the 15th International Joint Conference on Artifical Intelligence - Volume 1, IJCAI'97. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp 4–9. http://dl.acm.org/citation.cfm?id=1624162.1624163
Vincent P, Larochelle H, Bengio Y, Manzagol P-A (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. ICML '08. ACM, New York. pp 1096–1103. https://doi.org/10.1145/1390156.1390294. http://doi.acm.org/10.1145/1390156.1390294
Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A (2010) Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
Wilson RC, Zhu P (2008) A study of graph spectra for comparing graphs and trees. Pattern Recogn 41(9):2833–2841. https://doi.org/10.1016/j.patcog.2008.03.011
Wu G, Chang EY, Zhang Z (2005) An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. In: Proceedings of the 22nd International Conference on Machine Learning. International Conference on Machine Learning (ICML)
Xu K, Wu L, Wang Z, Feng Y, Witbrock M, Sheinin V (2018) Graph2seq: Graph to sequence learning with attention-based neural networks. In: arXiv Preprint arXiv:1804.00823

Yanardag P, Vishwanathan SVN (2015) Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '15. ACM, New York. pp 1365–1374. https://doi.org/10.1145/2783258.2783417. http://doi.acm.org/10.1145/2783258.2783417

## Publisher's Note