2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

A Simple Embedding for Classifying Networks with a few Graphlets

Luce le Gorrec
Department of Mathematics and Statistics
University of Strathclyde
Glasgow G1 1XH, United Kingdom
luce.le-gorrec@strath.ac.uk

Philip A. Knight
Department of Mathematics and Statistics
University of Strathclyde
Glasgow G1 1XH, United Kingdom
p.a.knight@strath.ac.uk

***Abstract*—Complex networks are a key analytical tool for complex systems. However if one wants to apply this tool in machine learning applications where the data is non-relational data one must find an appropriate network embedding. The embedding represents a network in a vector space, while preserving information about network structure. In this paper, we propose a simple network embedding technique that avoids the need for graph kernels or convolutional networks, as have previously been advocated. Our embedding is based on 3-node and 4-node graphlet counts combined with some feature extraction based on a Principal Component Analysis (PCA). We show that it is competitive with some state-of-the-art methods on a downstream classification task. We then show how to reduce the computational effort in the method by transforming extraction into a feature selection procedure. We claim that this selection procedure, a generalisation of PCA, is more meaningful than a popular alternative.**

***Index Terms*—Complex Networks, Graphlets, Graph Classification**

I. Introduction

Complex networks are an established tool for analysis in several scientific fields [1]. They can represent a complex system by modelling local interactions between elements within this system—for example, social relations among individuals [2], regulatory relations between genes [3], etc. The analysis of such networks built on these local interactions allows us to observe and understand phenomena at a larger scale [4]–[6].

A recent development in analysis has been fuelled by the observation that by representing networks as low-dimensional vectors one can then apply efficient machine learning algorithms [7]. Whilst most research has focused on finding fixed-length vectors to represent nodes of a network [8]–[10], techniques to embed the whole network as a unique prescribed-length vector have also appeared which allow comparison between sets of networks, often as a step towards classification [7], [11]–[13]. Reducing networks to these vector forms often involves non-trivial tools and may require a large amount of training data.

This project was supported by the Royal Academy of Engineering and the Office of the Chief Science Advisor for National Security under the UK Intelligence Community Postdoctoral Fellowship Programme.

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 Crown

Network embedding techniques share the desirable property that networks that share many substructures are close in the embedding space [12], [14]–[16]. This desirable property can be related to motifs in complex networks [17]–[22]. Indeed, it is well-known that induced subgraphs can have a functional meaning and can explain phenomena within the network. For instance, the sign-sensitive delay is closely related to a certain type of feed-forward loop in pathways [23]; and in neuronal networks, bifan motifs are known to cooperatively propagate information within synapses [24].

Strictly speaking motifs are small subgraphs (3 to 7 nodes) that appear significantly more often in a network of interest than in random networks sharing common properties. It has been shown that networks from a common field (e.g. pathways, neuronal networks, food webs, social networks, etc.) show similarities in 3- and 4-node motif distributions (also called significance profile or subgraph ratio profile (SRP)) [18]. The classical way to find motifs is to compute the number of occurrences of subgraphs in a sequence of explicitly generated random graphs and to compare these numbers with the number of occurrences of these subgraphs in the real-world network [17]–[19]. This technique cannot be applied to very large networks, as both generating the random graphs and counting the motif occurrences can be hugely expensive and time consuming. Moreover, there is no consensus as to the type of model that should be used to generate random graphs [13], [25], [26].

In this study, we propose a simple supervised technique for directed network embedding, based the distribution of k-node induced subgraphs, with $k = 3, 4$. By avoiding a comparison with random models it is much cheaper than classical SRP methods and despite its underlying simplicity it is highly effective as a tool for graph classification.

We summarise our contribution as follows. We propose a simple network embedding based on 3- and 4-node graphlets combined with dimension reduction procedure that is competitive with state-of-the-art methods on a downstream classification task. We propose a supervised feature selection procedure that allows one to focus on a handful of graphlets. This simple procedure is shown to outperform more complex widely-used feature selection technique. And we provide empirical evidence to support the claim that our supervised feature selection

TABLE I
NUMBER OF NODES (n) AND EDGES (m) OF THE NETWORKS.

	Food Webs		Elec. Circ.		Disc. Struct.		Soc. Net.	
	n	m	n	m	n	m	n	m
min	51	113	54	71	50	53	51	114
max	214	5,643	24097	52344	237	285	82168	870161
mean	104.4	1023	3358.9	6006.7	75.92	94.46	6430	48371

procedure can be used as an unsupervised tool, too. We also provide a new dataset for graph classification containing larger networks than is typical. Comprehensive details of this dataset can be found in [27]. In summary, we have extracted networks from four different fields, 70 representing food webs, 52 networks corresponding to electronic circuits, 195 discourse structure networks and 81 social networks. Summary statistics are provided in Table I.

II. RELATED WORK

The purpose of this study is to build network embeddings which are well-designed for a downstream classification task, which is the purpose of several existing techniques [7]. Broadly speaking, these techniques can be divided into three families, namely extensions of node embeddings, graph kernel methods, and graph neural networks.

Node embedding techniques aim to find embeddings for the nodes of one (or several) network(s) that are consistent with some similarity measure. For instance, one may want to consider nodes to be similar if they share an edge, neighbourhood or structural role and the chosen measure leads to a number of different embedding tools [8]–[10], [28]–[31]. Node embeddings can be used to induce network embeddings, for example using a weighted sum [7]. However, network representations based on postprocessing of learnt node embeddings may not be as effective as directly learnt network representations [12].

Rather than explicitly providing an embedding, graph kernels offer a direct method of measuring network similarity that can be interpreted in functional analysis terms [11], [15]. Methods differ in the means of capturing similarity [14], [15], [32] but we believe the technique closest to ours is [33], where the structure of networks is captured by computing the k-node graphlet occurrences (for $k \in \{3, 4, 5\}$), and where the embedding is explicitly provided. Key differences with our study are that the embeddings are not concatenated for the different values of k, and no feature reduction procedure is applied.

Automated machine learning methods can prove competitive with those based on handcrafted features. For instance, using spectral graph theory [16], Weisfeiler-Lehman kernels [9], [34], or grid-based node ordering [35], [36], a Convolutional Neural Network (CNN) can be adapted to a graph setting. Attention mechanisms have also been adapted to create Graph Attention Layers (GATs) [37]. These graph layers are naturally defined to provide one output per node, but can be used for graph classification by adding an aggregation or a pooling layer at the end of the neural network [38].

As for graph kernels, network embeddings often emerge as a side effect by processing the output prior to its synthesis for a given learning task (e.g. the label of the network for a classification task). To circumvent the common of dependence on the downstream learning task, the authors of `graph2vec` use task-agnostic embeddings [12], extending the idea behind the neural embedding `doc2vec` from natural language processing.

Finally, we mention `gl2vec` [13] which builds a network embedding by computing its 3-node SRP using a random model which preserves only the number of nodes and edges from the given network thereby reducing much of the computational cost of enumerating graphlets. While we share a focus on graphlets in our algorithm, we do not use a random model for comparison. Another fundamental difference is that `gl2vec` only uses 3-node graphlets and does not perform feature reduction. By performing such a reduction in combination with an enumeration of 4-node graphlets, our method outperforms `gl2vec`.

III. PROBLEM FORMULATION

To formulate our problem we provide some definitions to be used throughout the paper. Networks are assumed to be directed, unweighted without self-loops. The set of all such networks is denoted by $\mathcal{G}$. The network with nodes $V = \{v_1, \ldots, v_n\}$ and edges $E = \{(u, v) \in V \times V, u \neq v\}$ is denoted $G = (V, E)$. Given an integer k and the set

$$\mathcal{G}_k = \{G = (V, E) \in \mathcal{G} : |V| = k \text{ and } G \text{ is connected}\},$$

we will denote by k-node graphlets the set of non-isomorphic graphs in $\mathcal{G}_k$.

We denote by ϕ_k the function that counts the number of occurrences of induced k-node graphlets in a network, that is $\phi_k : \mathcal{G} \to \mathbb{R}_+^{m_k}$ with m_k the number k-node graphlets and the ith coordinate of $\phi_k(G)$ is the number of occurrences of the ith k-node graphlet in G.

Given vectors $\mathbf{v}_1 \in \mathbb{R}^{n_1}, \ldots, \mathbf{v}_p \in \mathbb{R}^{n_p}$, we use the term "concatenation" to mean the direct sum denoted by $\bigoplus_{i=1}^{p} \mathbf{v}_i$. This can also be written $(\mathbf{v}_1^T, \ldots, \mathbf{v}_p^T)^T \in \mathbb{R}^N$ with $N = \sum_{i=1}^{p} n_i$.

The aim of our study is to find an embedding of networks that fits with the classification task corresponding to labelling a network with the field it is extracted from. In the context of this study, a network embedding is a function $f : \mathcal{G} \to \mathbb{R}^d$. Formally, we have a dataset $\mathcal{D} = \{(G_i \in \mathcal{G}, \ell_i \in \mathcal{L})\}_{i \in \mathcal{I}}$, where $\mathcal{L}$ is the set of the labels (in our case, discrete labels corresponding to the field). The aim of a classification task is to find a function $g : \mathcal{G} \to \mathcal{L}$ called a classifier such that for (most) $i \in \mathcal{I}$, $g(G_i) = \ell_i$.

In the following, we will perform a supervised clustering for which the dataset $\mathcal{D} = \{(G_i \in \mathcal{G}, \ell_i \in \mathcal{L})\}_{i \in \mathcal{I}}$ is partitioned into a training set and a test set. The indices of elements in the training and test sets will be denoted by $\mathcal{S} \subset \mathcal{I}$ and $\mathcal{T} \subset \mathcal{I}$ respectively.

As the focus of our work is on the embedding we will consider the classifier separately, that is we extend g so that $g : \mathbb{R}^d \to \mathcal{L}$ whereby the whole classification process is then given by $g(f(G))$. Furthermore, we will use simple classifiers. For instance, we will make a great use of

$$g(\mathbf{x}) = \arg\min_{\ell \in \mathcal{L}} \|\mathbf{x} - \overline{\mathbf{x}}_\ell\|_2 \tag{III.1}$$

where $\overline{\mathbf{x}}_\ell = \dfrac{1}{|\{i \in \mathcal{S} \text{ s.t. } \ell_i = \ell\}|} \displaystyle\sum_{i \in \mathcal{S}: \ell_i = \ell} f(G_i)$.

IV. Naive Embedding and Classification Task

Our aim is to find an embedding of networks based on 3-node and 4-node graphlet counts. We start by considering a naive embedding $f : \mathcal{G} \to \mathbb{R}^M$, with $M = m_3 + m_4 = 212$, such that, $\forall G \in \mathcal{G}$:

$$f(G) = \frac{\bigoplus_{k=3,4} \alpha_k(G)\phi_k(G)}{\sqrt{\sum_{k=3,4} \alpha_k(G)^2 \phi_k(G)^T \phi_k(G)}} \tag{IV.1}$$

where $\alpha_k(G)$ are positive real numbers used to mitigate the order of magnitude difference between $\phi_3(G)$ and $\phi_4(G)$. We use $\alpha_k(G) = \dfrac{m_k}{k}\binom{k}{n}^{-1}$, which is proportional to the ratio between the number of k-node graphlets and the maximum number of such graphlets that can occur in a network of n nodes. The normalisation of $f(G)$ makes comparison between networks more straightforward.

As the dimension of the embedding produced by f is high (higher than the number of networks in the training set), we combine the classifier with a feature extraction procedure based on PCA, by following the steps proposed in [39]. That is, we compute a matrix $\widetilde{\mathbf{U}}$ containing the leading eigenvectors of the covariance matrix of the training set, and we use $\widetilde{\mathbf{U}}$ to get the shifted projections of samples from both training and test sets onto a lower dimension space whose directions are the principal directions provided by $\widetilde{\mathbf{U}}$. Then we apply the classifier from (III.1) to these embeddings. The complete process are provided in Algorithms 1 and 2, where $\mathbf{e}_p \in \mathbb{R}^p$ is a vector of ones.

Algorithm 1: Naive Classification

Input: $G \in \mathcal{G}$, $\widetilde{\mathbf{U}}^k \in \mathbb{R}^{M \times k}$, $\overline{\mathbf{x}} \in \mathbb{R}^M$, $\mathcal{C} = \{(\overline{\mathbf{c}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \ldots, |\mathcal{L}|\}$
Output: $\ell^* \in \mathcal{L}$ the predicted class of G

begin
  compute $\phi_3(G)$, $\phi_4(G)$, $\alpha_3(G)$, $\alpha_4(G)$
  $\mathbf{x} \leftarrow f(G)$ according to (IV.1)
  $\mathbf{c} \leftarrow (\widetilde{\mathbf{U}}^k)^T \times (\mathbf{x} - \overline{\mathbf{x}})$
  $(\mathbf{c}^*, \ell^*) = \arg\min_{(\overline{\mathbf{c}_i}, \ell_i) \in \mathcal{C}} \|\mathbf{c} - \overline{\mathbf{c}_i}\|_2$
return ℓ^*

Algorithm 2: Preprocessing of Algorithm 1

Input: A training set $\{(G_\sigma, \ell_\sigma) \in \mathcal{G} \times \mathcal{L}, \forall \sigma \in \mathcal{S}\}$, an integer $k < M$
Output: $\overline{\mathbf{x}} \in \mathbb{R}^M$, $\widetilde{\mathbf{U}}^k \in \mathbb{R}^{M \times k}$, $\mathcal{C} = \{(\overline{\mathbf{c}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \ldots, |\mathcal{L}|\}$

begin
  for $\sigma \in \mathcal{S}$ **do**
    compute $\phi_3(G_\sigma), \phi_4(G_\sigma), \alpha_3(G_\sigma), \alpha_4(G_\sigma)$
    $\mathbf{x}_\sigma \leftarrow f(G_\sigma)$ according to (IV.1)
  $\overline{\mathbf{x}} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\sigma \in \mathcal{S}} \mathbf{x}_\sigma$
  apply PCA on $\mathbf{X} = \begin{bmatrix} \mathbf{x}_{\sigma_1} & \ldots & \mathbf{x}_{\sigma_{|\mathcal{S}|}} \end{bmatrix}^T$ to obtain $\mathbf{U} \in \mathbb{R}^{M \times M}$
  $\widetilde{\mathbf{U}}^k \leftarrow \mathbf{U} \times \begin{bmatrix} \mathbf{I}_k \\ 0 \end{bmatrix}$; $\mathbf{C} \leftarrow (\mathbf{X} - \mathbf{e}_{|\mathcal{S}|} \times \overline{\mathbf{x}}^T) \times \widetilde{\mathbf{U}}^k$
  $\mathcal{C} \leftarrow \emptyset$
  for $\ell \in \mathcal{L}$ **do**
    $\overline{\mathbf{c}_\ell} \leftarrow \frac{1}{|\{\sigma \in \mathcal{S}: \ell_\sigma = \ell\}|} \sum_{\sigma \in \mathcal{S}: \ell_\sigma = l} \mathbf{c}_\sigma$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\overline{\mathbf{c}_\ell}, \ell)\}$
return $\overline{\mathbf{x}}$, $\widetilde{\mathbf{U}}^k$, $\mathcal{C}$

TABLE II
Quality measurement of embeddings.

		Precisiom		Recall		F1-score	
		mean	std	mean	std	mean	std
our method	*fw*	9.50	2.6	**10**	0.3	**9.74**	1.4
	elec	10.0	0.3	9.00	9.0	9.45	5.0
	disc	9.88	0.8	9.95	0.3	9.91	0.4
	soc	**10**	0.2	9.63	2.1	**9.81**	1.1
gl2vec	*fw*	**9.76**	2.5	8.65	5.3	9.16	3.3
	elec	9.26	8.5	6.84	11.2	7.79	8.2
	disc	9.89	0.7	**10**	0	9.94	0.4
	soc	8.98	3.2	**10**	0	9.46	1.8
graph 2vec	*fw*	9.51	3.6	9.73	3.1	9.61	2.6
	elec	**10**	0	**9.80**	3.6	**9.90**	1.9
	disc	**9.90**	0.7	**10**	0	**9.95**	0.3
	soc	9.80	2.3	9.30	3.9	9.54	2.4

A. Numerical Experiments

In order to assess the accuracy of the method presented in Algorithm 1, we have applied it to recover the labels of the networks in our dataset. The training set $\mathcal{S}$ is built by randomly selecting $N_B = 40$ networks from each class/field. Thus, $|\mathcal{S}| = |\mathcal{L}| \times N_B$. The remaining form the test set $\mathcal{T}$, on which Algorithm 1 is applied[1]. We set the number of principal axes to keep to be $k = 11$—justification is detailed in [27]. To compute ϕ_3 and ϕ_4 in (IV.1), we have used `acc-Motif` [40][2].

We compare the quality of our embeddings with those obtained by `gl2vec` and `graph2vec`, using the implementations from [41] and [42] respectively. Several sets of

[1] 500 pairs of sets $(\mathcal{S}, \mathcal{T})$ were built by randomly selecting N_B networks from each class. We present the mean and standard deviation of the results.

[2] version 2.2, available at http://www.ft.unicamp.br/docentes/meira/accmotifs/

parameters have been tested for `graph2vec`, and the results presented here are the most accurate we obtained, achieved with a depth of 1 for Weisfeiler-Lehman (WL) kernel, and a dimension of 64 for the embeddings. Precise settings and parameter choices for these algorithms are detailed in [27]. The results of the classifier of (III.1) applied to these three different embeddings are provided in Table II. The labels *fw*, *elec*, *disc* and *soc* relate to respectively the food web, electronic circuit, discourse structure and social relationship benchmarks. For convenience of presentation, the means and standard deviations have been multiplied respectively by 10^{-1} and 10^{-2} in the table.

We observe that all embedding techniques return consistent clusters for each class. With the exception of electronic circuit networks embedded via `gl2vec`, all $F1$-score are above 0.9. We remark that `graph2vec` outperforms `gl2vec` as well as our method on the discourse structure networks (for which the gain is slight as all methods achieve very good $F1$-score for this benchmark), and on electronic circuits (where the gain is indeed quite substantial). On the other hand, our method significantly outperforms `gl2vec` and `graph2vec` on food webs and social networks. We observe that these last two classes are less well defined in the sense that they are built from observations.

B. Runtime

In this section, we briefly discuss the runtimes of the different methods. All the experiments were conducted on a laptop with 32GB RAM and a Core i5 vPro 8th Generation as a processor.

In the case of our method, the most time-consuming part is the calls to `acc-Motif` to compute the graphlet distributions, which takes around 15min. However, our calls to `acc-Motif` are done sequentially, and a parallelisation could greatly boost it, since the maximum time for one call is 116s[3].

As a matter of comparison, the computation of the SRPs in $gl2vec$ takes about 3min. It is worth to say that, as a random model for computing the SRPs, we have chosen the only one that does not require to explicitly build random networks, as otherwise the runtimes were prohibitive (we stopped the execution after 30min).

Finally, computing the embedding with $graph2vec$ lasts 2min: 1min for building the node features, and 1min to compute the 64-dimension embeddings via a WL-kernel of dimension 1. We remark that, for deeper WL-kernels (1 being the smallest possible value), the runtimes naturally increase (80s for a depth of 2, 100s for a depth of 3).

The training part of our method, that consist in extracting the dominant eigenvectors of the covariance matrix of the training set, takes less than 4s to be executed for 500 iterations. For the 3 methods, the classification part is neglectable (between 1s and 2s for the 500 iterations, $graph2vec$ being the slowest as the dimension of embeddings are higher).

[3]For computing the 4-node graphlet distribution of a 53K node network.

V. Graphlet Selection

We have established that it is possible to build a naive graph embedding based on graphlet counting coupled with feature extraction that provides accurate graph classification. This simple method is competitive with recently proposed embedding techniques that outperform classical state of the art methods in the specific downstream task of graph classification.

Nevertheless, the naive procedure presented in Algorithm 1 is suboptimal. Indeed, it relies on an expensive count of all 3-node and 4-node graphlets for each network which we believe is unnecessary. Furthermore, the construction of the projection matrix $\widetilde{\mathbf{U}}^k$ in Algorithm 1 strongly depends on the training set. To avoid these drawbacks, we propose to perform feature selection instead of feature extraction so that we can avoid both the full computation of graphlets as well as the projection using $\widetilde{\mathbf{U}}^k$.

We build our procedure for feature selection by drawing parallels with some basic principles of PCA. We briefly describe our ideas below. Detailed explanations can be found in [27]. First note that given a dataset $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \dots & \mathbf{x}_n\end{bmatrix}^T \in \mathbb{R}^{n\times M}$, in which each row is a sample, and where $\overline{\mathbf{x}}$ is the centre of gravity (i.e. the average sample), the projection in PCA can be thought of in terms of optimisation in that for a given value of q, $\widetilde{\mathbf{U}}^q$ solves

$$\begin{cases} \underset{\mathbf{U}\in\mathbb{R}^{M\times q}}{\arg\max} & \mathcal{I}_{\mathbf{U}}(\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{y}_i - \overline{\mathbf{y}}\|_2^2 \\ \text{with} & \mathbf{y}_i = \mathbf{U}\mathbf{U}^T(\mathbf{x}_i - \overline{\mathbf{x}}) + \overline{\mathbf{x}}, \\ \text{and} & \overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i \\ \text{subject to} & \mathbf{U}^T \times \mathbf{U} = \mathbf{I}_q, \end{cases} \tag{V.1}$$

where $\mathcal{I}_{\mathbf{U}}(\mathbf{X})$, called the inertia, measures the deviation from the mean of the projections of the rows of $\mathbf{X}$ into the affine space that is the translation of the vector space with basis $\mathbf{U}$ that passes through $\overline{\mathbf{x}}$. We can also cast the matrix of these projections $\mathbf{Y}$, with rows $\mathbf{y}_i^T$, as the minimiser of an approximation error problem.

Our aim is to build a feature selection procedure—that is to pick k vectors from the canonical basis—following similar principles to PCA. We assume that we know $\mathbf{Y} \in \mathbb{R}^{n\times M}$ that solves (V.1). We design our feature selection procedure as a maximisation of an inertia problem. To be precise, we find $\widetilde{\mathbf{E}}^k \in \mathbb{R}^{M\times k}$ that solves

$$\mathcal{Q}_k : \begin{cases} \underset{\mathbf{E}\in\mathbb{R}^{M\times k}}{\arg\max} & \mathcal{I}_{\mathbf{E}}(\mathbf{Y}) = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{z}_i - \overline{\mathbf{z}}\|_2^2 \\ \text{with} & \mathbf{z}_i = \mathbf{E}\mathbf{E}^T(\mathbf{y}_i - \overline{\mathbf{y}}) + \overline{\mathbf{y}}, \\ \text{and} & \overline{\mathbf{z}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{z}_i \\ \text{subject to} & \mathbf{E} = \begin{bmatrix}\mathbf{e}_{\sigma_1} & \dots & \mathbf{e}_{\sigma_k},\end{bmatrix} \end{cases} \tag{V.2}$$

where $\mathbf{e}_j$ is the jth column of the identity matrix.

As for PCA, $\widetilde{\mathbf{E}}^k$ defines the basis of an affine space in which projections of rows of $\mathbf{Y}$ also minimise the approximation error. Furthermore, $\mathcal{Q}_{k+1}$ can be written as $\widehat{\mathbf{E}} = \left[\begin{array}{c|c} \widetilde{\mathbf{E}}^k & \mathbf{e}_{\sigma_{k+1}} \end{array}\right]$. The proofs of these assertions are fairly long and can be found in [27]. A consequence is that, as for

PCA in which each column of $\mathbf{U}$ adds one eigenvalue of the covariance matrix of $\mathbf{X}$ to the inertia $\mathcal{I}_{\mathbf{U}}(\mathbf{X})$, it is possible to write the global inertia $\mathcal{I}_{\widetilde{\mathbf{E}}^k}(\mathbf{Y})$ as the sum of the individual contributions to the inertia from each canonical axis. This enables us to derive a score that assesses the significance of a canonical axis, given the q first principal axes.

Definition 1. *The expression level of the ith canonical axis within the q first principal axes is defined to be*

$$\gamma(i) = \frac{1}{\sum_{s=1}^{q} \lambda_s} \sum_{t=1}^{q} \lambda_t \mathbf{u}_t(i)^2. \tag{V.3}$$

The measure $\gamma(i)$ gives an indication of the significance of the ith canonical axis, and is normalised so that $\sum_{i=1}^{k} \gamma(i) = 1$. It allows one to derive a feature selection procedure for networks in the test set: knowing the q principal axes of the training set, one can choose the k canonical axes with the highest γ-score to embed the networks from the test set. We arrive at a set $\Gamma \subset \{1, \dots, M\}, |\Gamma| = k$, of indicators of the best canonical axes to keep.

To derive our embedding function, we separate indicators of 3-node graphlets and 4-node graphlets, that is we build $\Gamma_3 \subset \{1, \dots, m_3\}$ and $\Gamma_4 \subset \{1, \dots, m_4\}$ such that $\Gamma_3 \cup (\Gamma_4 + m_3) = \Gamma$ (with $\Gamma_4 + m_3 = \{\sigma_i + m_3, \sigma_i \in \Gamma_4\}$. Our embedding function is

$$h(G) = \frac{\bigoplus_{t=3,4} \alpha_t(G) \psi_t(G, \Gamma_t)}{\sqrt{\sum_{t=3,4} \alpha_t(G)^2 \psi_t(G, \Gamma_t)^T \psi_t(G, \Gamma_t)}}, \tag{V.4}$$

with $\alpha_t(G)$ as defined in (IV.1), and

$$\psi_t(G, \Gamma_t) = \mathbf{P}_t \times \phi_t(G),$$

with

$$\mathbf{P}_t \in \mathbb{R}^{|\Gamma_t| \times m_t} \text{ s.t. } \mathbf{P}_t(i,j) = \begin{cases} 1 & \text{if } \Gamma_t(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

Although in the previous equation, we assume that we extract the selected features from the whole 3-and 4-node graphlet decomposition, it is actually possible to compute only the number of occurrences of the graphlets in Γ, which may be computationally cheaper—see [43] for a discussion on the different methods to decompose a network partially or entirely into motifs, and their complexity.

The γ-measure we defined above allows one to simplify the classification algorithm presented in Section IV. Below we propose Algorithm 3, an adaptation of Algorithm 1 based on this γ-measure, and Algorithm 4, which is the preprocessing of Algorithm 3.

Algorithm 3: Classification with Feature Selection

Input: $G \in \mathcal{G}$,
$\Gamma_3 \subset \{1, \dots, m_3\}$, $\Gamma_4 \subset \{1, \dots, m_4\}$,
$\mathcal{C} = \{(\overline{\mathbf{z}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \dots, |\mathcal{L}|\}$
Output: $\ell^* \in \mathcal{L}$ the predicted class of G

begin
  compute $\psi_3(G, \Gamma_3)$, $\psi_4(G, \Gamma_4), \alpha_3(G), \alpha_4(G)$
  $\mathbf{z} \leftarrow h(G)$ according to (V.4)
  $(\mathbf{z}^*, \ell^*) = \underset{(\overline{\mathbf{z}_i}, \ell_i) \in \mathcal{C}}{\arg\min} \|\mathbf{z} - \overline{\mathbf{z}_i}\|_2$
return ℓ^*

Algorithm 4: Preprocessing of Algorithm 3

Input: A training set $\{(G_\sigma, \ell_\sigma) \in \mathcal{G} \times \mathcal{L}, \forall \sigma \in \mathcal{S}\}$,
two integers $k, q < M$
Output: $\mathcal{C} = \{(\overline{\mathbf{z}_i}, \ell_i) \in \mathbb{R}^k \times \mathcal{L}, i = 1, \dots, |\mathcal{L}|\}$,
$\Gamma_3 \subset \{1, \dots, m_3\}$, $\Gamma_4 \subset \{1, \dots, m_4\}$

begin
  for $\sigma \in \mathcal{S}$ **do**
    compute $\phi_3(G_\sigma), \phi_4(G_\sigma), \alpha_3(G_\sigma), \alpha_4(G_\sigma)$
    $\mathbf{x}_\sigma \leftarrow f(G_\sigma)$ according to (IV.1)
  $\overline{\mathbf{x}} \leftarrow \frac{1}{|\mathcal{S}|} \sum_{\sigma \in \mathcal{S}} \mathbf{x}_\sigma$
  apply PCA on $\mathbf{X} = \begin{bmatrix} \mathbf{x}_{\sigma_1} & \dots & \mathbf{x}_{\sigma_{|\mathcal{S}|}} \end{bmatrix}^T$ to obtain the q ppal. axes $\mathbf{U} \in \mathbb{R}^{M \times q}$
  $\Gamma \leftarrow \mathbf{0}_{k \times 1}$, $\Lambda \leftarrow \mathbf{0}_{k \times 1}$
  for $t = 1, \dots, M$ **do**
    if $\gamma(t) > \min(\Lambda)$ **then**
      $i^* \leftarrow \underset{i=1,\dots,k}{\arg\min}(\Lambda(i))$
      $\Lambda(i^*) \leftarrow \gamma(t)$
      $\Gamma(i^*) \leftarrow t$
  $\Gamma_3 \leftarrow \{t \in \Gamma : t \leq m_3\}$
  $\Gamma_4 \leftarrow \{t - m_3, \forall t \in \Gamma : t > m_3\}$
  for $\sigma \in \mathcal{S}$ **do**
    $\mathbf{z}_\sigma \leftarrow h(G_\sigma)$ according to (V.4)
  $\mathcal{C} \leftarrow \emptyset$
  for $\ell \in \mathcal{L}$ **do**
    $\overline{\mathbf{z}_\ell} \leftarrow \frac{1}{|\{\sigma \in \mathcal{S} : \ell_\sigma = \ell\}|} \sum_{\sigma \in \mathcal{S} : \ell_\sigma = l} \mathbf{z}_\sigma$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{(\overline{\mathbf{z}_\ell}, \ell)\}$
return Γ_3, Γ_4, $\mathcal{C}$

A. Numerical Experiments

We test the enhanced Algorithms 4 and 3 on our dataset as before. That is, by randomly selecting N_B networks from each field to build the training set $\mathcal{S}$ and use the remaining networks as test set $\mathcal{T}$. Again, we perform 500 realisations and we present the mean results and their standard deviation. We fix the number of principal axes used to $q = 11$ as previously.

To measure the consistency of the γ-scores of graphlets over the 500 training sets we plot the error bar of the 20 graphlets with highest average γ-score in the top plot of Fig. 1. The vertical bars indicate a range of plus or minus one standard deviation around the average of each γ-score of a graphlet. Identifiers of graphlets are provided on the x-axis. Details about these identifiers are given [27].

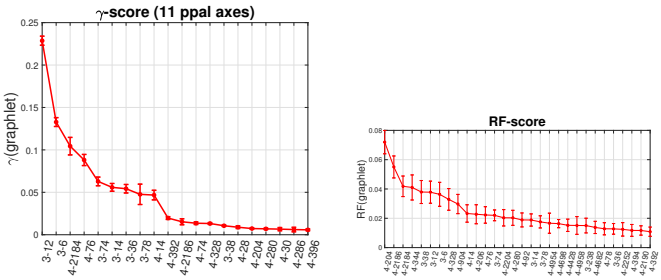

Fig. 1. Left: The 20 graphlets with the highest average γ-score. Right: The 30 graphlets with the highest score returned by Random Forest.

TABLE III
QUALITY MEASUREMENT OF THE EMBEDDINGS CLASSIFIED BY (III.1).

		Precision		Recall		F1-score	
		mean	std	mean	std	mean	std
our method	fw	9.50	2.7	9.78	2.6	9.64	1.9
	$elec$	9.44	6.3	8.86	8.4	9.11	5.8
	$disc$	9.86	0.8	9.95	0.3	9.90	0.4
	soc	10	0	9.61	2.5	9.80	1.3
RF feat. select.	fw	9.17	4.0	8.74	6.7	8.93	4.0
	$elec$	4.30	12.3	7.77	18.2	5.46	12.8
	$disc$	9.70	2.2	9.77	3.7	9.73	2.4
	soc	8.16	9.5	6.08	13.0	6.91	11.0

One can see that graphlet 3-12 (namely ) has by far the highest score, accounting for almost a quarter of the inertia in the space of the q principal axes. There is a noticeable drop in the curve between the ninth and tenth scores and so we have chosen to keep only the 9 graphlets with highest average γ-score.

The accuracy of the classifier produced by Algorithm 3 with feature selection using (V.4) is demonstrated in Table III. As a comparison, we have built input subsets Γ_3 and Γ_4 using a feature selection based on random forests (RF) [44] and then applied Algorithm 3, as detailed in [27]. This implementation allows one to get a score of importance for each feature after having trained the RF model, which is the average Gini importance over all trees. For a fair comparison, we have kept the 9 graphlets with highest Gini importance.

Results are presented in Table III. As before, the displayed mean and standard deviation values have been scaled to ease presentation. It is clear that selecting features based on γ-score provides far better results than using RF based feature selection. Our new method is significantly better both in terms of precision and recall, for every label. Moreover, we can see in the bottom plot of Fig. 1, that the shape of the curve of Gini importance over graphlets is much less sharp at the cut-off point than the γ-score curve.

Finally, by comparing Tables II and III, we observe that Algorithm 3 produces similar results to Algorithm 1 at lower cost. The conclusions of the previous section remain true: our new method is more accurate than `gl2vec` and `graph2vec` in uncovering food webs, and social networks. The results for the rhetorical discourse benchmark are still high ($F1$-score = 0.99). With feature selection, there is a noticeable decrease of accuracy in identifying electronic circuits. Nevertheless, even with this decrease it remains significantly more accurate than `gl2vec` ($F1$-score = 0.779).

VI. TOWARDS UNSUPERVISED CLUSTERING

In the previous section, we proposed a network embedding based on 3-node and 4-node graphlets decomposition, followed by a feature selection procedure built on a training set of networks. We demonstrated that this method is very efficient in terms of network classification. We predict that the graphlets we identified in Section V-A should also be able to discriminate other classes of networks. That is, our selection of the 9 graphlets with highest average γ-score (see Figure 1) may be a good choice to perform feature selection in an unsupervised fashion.

To support this claim we have applied the embedding given in (V.4), with $\Gamma_3 = \{3\text{-}12,3\text{-}6,3\text{-}74,3\text{-}14,3\text{-}36,3\text{-}78\}$ and $\Gamma_4 = \{4\text{-}2184,4\text{-}76,4\text{-}14\}$ to the whole dataset supplemented with 60 directed networks built using the graph generator from [45]. This generator builds random modular networks with features observed in real-world networks (such as a heterogeneous distributions of node degrees and community sizes described by power laws). Several parameters need to be set up and we have built the networks with a mixture of bespoke and default values, which we describe in [27].

To uncover the network clusters, we have applied the unsupervised clustering algorithm from [46], which requires no prior knowledge of the number of clusters. This algorithm needs the dataset to be in an affinity matrix style[4]. To do this here we have used the affinity matrix of our embedded networks $\mathbf{x}_i, i \in \mathcal{I}$ built with (V.4). The affinity matrix of a set of points $\{\mathbf{x}_i \in \mathbb{R}^p, i = 1,\dots,n\}$ being the symmetric matrix with zeros on the diagonal and

$$\mathbf{A}(i,j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right),$$

elsewhere. The affinity matrix strongly depends on the choice of the Gaussian parameter σ. We follow the prescription in [47] to take into account both density and dimension of the dataset. We have then sparsified the affinity matrix by removing all the entries below a certain threshold (here 0.5, which removes 65% nonzero entries). Moreover, as shown in Fig. 2, the affinity between network embeddings is not homogeneous through the different clusters, and we have thus applied another sparsification for each network by removing all its affinities except those of its 20 closest neighbours. This results in a nonsymmetric matrix that the algorithm from [46] is able to handle.

The unsupervised clustering method applied to both sparsified matrices gives consistent results. For the first, it finds the 5 targeted clusters with a few errors of assignation. For the second method, it finds 10 clusters, but these clusters are actual partitionings of the initial classes, and by merging together the clusters that belong to a same class (as highlighted by the

[4]That is, a dataset containing n samples has to be represented by a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ where $\mathbf{A}(i,j)$ expresses the proximity (or affinity) between sample i and sample j.

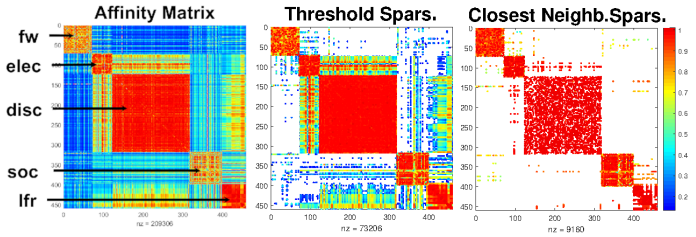

Fig. 2. Affinity matrix of the dataset, and its two sparsifications.

TABLE IV
TOP: CONFUSION MATRICES OF THE RESULTING CLUSTERING. LEFT: USING THE THRESHOLD SPARSIFICATION. RIGHT: USING THE CLOSEST NEIGHBOUR SPARSIFICATION. BOTTOM: RESULTING SCORES.

	1	2	3	4	5
fw	70				
$elec$			6		46
$disc$	1		192	2	
soc	1	75		5	
lfr				60	

	1	2	3	4	5	6	7	8	9	10
fw										70
$elec$			4	2					46	
$disc$			52	26	56	59		1		1
soc	40	38					1	1		1
lfr							10	50		

	Prec.	Rec.	F1-score
fw	9.72	10	9.86
$elec$	10	8.85	9.39
$disc$	9.70	9.85	9.77
soc	10	9.26	9.62
lfr	89.6	10	9.45

	Prec.	Rec.	F1-score
fw	9.72	10	9.86
$elec$	10	8.85	9.39
$disc$	9.70	9.90	9.80
soc	10	9.63	9.81
lfr	9.52	10	9.76

column colours in Table IV), the accuracy of the method is equal or higher than the first sparsification, for every class. The confusion matrices of the resulting clusterings are shown in Table IV, as well as the resulting scores in terms of precision, recall and $F1$-score.

These preliminary results strongly suggest that the 9 graphlets that discriminate between the 4 classes of networks in our numerical experiments from Section V-A are also able to discriminate other classes of networks. Nevertheless, these are early-stage experiments and this hypothesis needs to be tested on other classes of real-world networks to get a better feel for the extent to which these graphlets are sufficient to discriminate between classes.

VII. CONCLUSIONS

We have proposed two supervised procedures for network embedding based on 3-node and 4-node graphlet decomposition. These procedures have been shown to provide extremely accurate results in a downstream classification task. The whole process is based on very simple tools such as PCA, and is yet competitive with state-of-the-art methods. In contrast to most existing methods for classifying or comparing networks based on graphlet decomposition [13], [18], we avoid an evaluation of the graphlet distribution on random models. Of particular note is that our feature selection method selects only a few graphlets (9 out of a possible 212), which means we can avoid computing all graphlets for the networks in the test set. Our bespoke feature selection has properties similar to those of PCA, and we have shown that this procedure outperforms a well-established selection feature method.

TABLE V
ALGORITHM 1 APPLIED ON MUTAG DATASET.

	mutagenic effect	Precision mean	Precision std	Recall mean	Recall std	F1-score mean	F1-score std
our	no	9.24	2.1	8.55	3.2	8.88	1.8
method	yes	5.82	5.2	7.38	7.9	6.49	4.8
$graph$	no	8.83	2.2	8.60	4.2	8.70	2.2
$2vec$	yes	5.33	6.7	5.77	9.7	5.49	6.2
$gl2$	no	9.33	2.5	9.11	3.9	9.21	1.8
vec	yes	7.06	7.4	7.55	9.9	7.23	5.2

Finally, we have provided some early-stage empirical experiments that suggest that the graphlets selected by our feature selection procedure in the numerical experiments are actually able to discriminate other kinds of networks apart from those classes on which these graphlets have been learnt. We infer that these graphlets are sufficient to discriminate among wider classes of networks, and should enable one to derive unsupervised methods for network embedding based on just these few graphlets.

Our future work will be to extend these experiments to understand the full potential (as well as the limitations) of these graphlets to characterise fields of networks in order to design unsupervised techniques for network embedding.

Another aspect we would like to investigate is any gain of accuracy that may be brought by larger graphlets. Here we have limited ourselves to 3-node and 4-node graphlets, which was sufficient to accurately discriminate our networks through the numerical experiments, but our study and algorithms can be generalised to more complex graphlet decomposition straightforwardly.

Finally, our method can be applied to other applications of network classification, and non-directed networks as well. As a matter of illustration, we report in Table V the results of the classifier presented in Algorithm 1, applied on the $MUTAG$ dataset [48]. This classic benchmark contains 188 networks representing chemical compounds that one aims to classify onto compounds having a mutagenic effect on a bacteria, and compounds that have not. Once again, we compare our method to $graph2vec$[5] and $gl2vec$, and observe that we are competitive. Interestingly enough, this is $gl2vec$ that provides the best results, probably because the small size of networks (17 nodes in average) allows the 3-node graphlet distribution to closely fits with network structures.

Tables and Figures from this paper can be reproduced by using the Matlab code and dataset at http://github.com/luleg/DiscriminantMotifs, where networks used to conduct our experiments can also be found.

ACKNOWLEDGMENT

We would like to thank Kun Tu from the University of Massachusetts Amherst, for providing us with the code of `gl2vec` and answering our questions.

[5]The set of parameters achieving the highest scores for $graph2vec$ have been chosen, namely 1 for the depth of the WL kernel and 16 for the embedding dimension.

REFERENCES

[1] E. Estrada and P. Knight, *A First course in network theory*. Oxford University Press, 2015.

[2] P. Sapiezynski, A. Stopczynski, D. Lassen, and S. Lehmann, "Interaction data from the Copenhagen Networks Study," *Scientific Data*, vol. 6, no. 1, 2019.

[3] B. He and K. Tan, "Understanding transcriptional regulatory networks using computational models," *Current Opinion in Genetics and Development*, vol. 37, pp. 101–108, 2016.

[4] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *ACM Workshop on Online Social Networks*, p. 37–42, 2009.

[5] F. Gargiulo, A. Caen, R. Lambiotte, and T. Carletti, "The Classical origin of modern mathematics," *EPJ Data Science*, vol. 5:26, 2016.

[6] R. F. i Cancho, C. Janssen, and R. V. Solé, "Topology of technology graphs: Small world patterns in electronic circuits," *Physical Review E*, vol. 64, no. 4, p. 046119, 2001.

[7] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, 2017.

[8] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 855–864, 2016.

[9] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *ACM SIGKDD International Conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[11] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *The Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010.

[12] R. V. L. C. Y. L. Annamalai Narayanan, Mahinthan Chandramohan and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," in *International Workshop on Mining and Learning with Graphs*, 2017.

[13] K. Tu, J. Li, D. Towsley, D. Braines, and L. D. Turner, "Gl2vec: Learning feature representation using graphlets for directed networks," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, p. 216–221, 2019.

[14] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels.," *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[15] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning theory and kernel machines*, pp. 129–143, 2003.

[16] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[17] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," *Science*, vol. 298, no. 5594, pp. 824–827, 2002.

[18] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, "Superfamilies of evolved and designed networks," *Science*, vol. 303, no. 5663, pp. 1538–1542, 2004.

[19] D. Felmlee, C. McMillan, D. Towsley, and R. Whitaker, "Social network motifs: A comparison of building blocks across multiple social networks," in *Annual Meetings of the ASA*, 2018.

[20] M. Mesgar and M. Strube, "Graph-based coherence modeling for assessing readability," in *Joint conference on lexical and computational semantics*, pp. 309–318, 2015.

[21] N. T. L. Tran, S. Mohan, Z. Xu, and C.-H. Huang, "Current innovations and future challenges of network motif detection," *Briefings in bioinformatics*, vol. 16, no. 3, pp. 497–525, 2015.

[22] D. B. Stouffer, J. Camacho, W. Jiang, and L. A. Nunes Amaral, "Evidence for the existence of a robust pattern of prey selection in food webs," *Proceedings of the Royal Society B: Biological Sciences*, vol. 274, no. 1621, pp. 1931–1940, 2007.

[23] U. Alon, "Network motifs: theory and experimental approaches," *Nature Reviews Genetics*, vol. 8, no. 6, pp. 450–461, 2007.

[24] A. Benson, D. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[25] R. Milo, N. Kashtan, S. Itzkovitz, M. E. Newman, and U. Alon, "On the uniform generation of random graphs with prescribed degree sequences," *arXiv preprint*, 2003.

[26] Y. Artzy-Randrup, S. Fleishman, N. Ben-Tal, and L. Stone, "Comment on "network motifs: simple building blocks of complex networks" and "superfamilies of evolved and designed networks"," *Science*, vol. 305, no. 5687, pp. 1107–1107, 2004.

[27] L. le Gorrec and P. A. Knight, "Supplementary material for "a simple embedding for classifying networks with a few graphlets." http://github.com/luleg/DiscriminantMotifs, 2020.

[28] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *International Conference on World Wide Web*, p. 37–48, 2013.

[29] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 1105–1114, 2016.

[30] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385–394, 2017.

[31] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *AAAI Conference on Artificial Intelligence*, p. 1145–1152, 2016.

[32] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *IEEE International Conference on Data Mining*, p. 8 pp, 2005.

[33] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *International Conference on Artificial Intelligence and Statistics*, pp. 488–495, 2009.

[34] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[35] H. Peng, J. Li, Q. Gong, S. Wang, Y. Ning, and P. S. Yu, "Graph convolutional neural networks via motif-based attention," *arXiv:1811.08270*, 2018.

[36] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, pp. 2014–2023, 2016.

[37] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations*, 2018.

[38] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *International Conference on Learning Representations*, 2019.

[39] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[40] L. A. Meira, V. R. Máximo, Á. L. Fazenda, and A. F. Da Conceição, "Acc-motif: accelerated network motif detection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 11, no. 5, pp. 853–862, 2014.

[41] K. Tu. https://github.com/kuntu/JGraphlet-JMotif. Access: June 2020.

[42] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs," in *ACM International on Conference on Information and Knowledge Management*, 2020.

[43] P. Ribeiro, P. Paredes, M. E. Silva, D. Aparicio, and F. Silva, "A survey on subgraph counting: concepts, algorithms and applications to network motifs and graphlets," *arXiv:1910.13011*, 2019.

[44] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[45] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E*, vol. 80, p. 016118, Jul 2009.

[46] L. le Gorrec, S. Mouysset, I. S. Duff, P. A. Knight, and D. Ruiz, "Uncovering hidden block structure for clustering," in *Machine Learning and Knowledge Discovery in Databases*, pp. 140–155, 2020.

[47] S. Mouysset, J. Noailles, and D. Ruiz, "Using a global parameter for gaussian affinity matrices in spectral clustering," in *VECPAR*, vol. 5336 of *Lecture Notes in Computer Science*, pp. 378–390, Springer, 2008.

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