Knowledge and Information Systems (2023) 65:2455–2483
https://doi.org/10.1007/s10115-023-01836-3

REGULAR PAPER

Density of states for fast embedding node-attributed graphs

Lingxiao Zhao[1] · Saurabh Sawlani[2] · Leman Akoglu[1]

Received: 27 January 2022 / Revised: 17 January 2023 / Accepted: 18 January 2023 /
Published online: 20 February 2023

Abstract
Given a node-attributed graph, how can we efficiently represent it with few numerical features that expressively reflect its topology and attribute information? We propose A- DOGE, for attributed DOS-based graph embedding, based on density of states (DOS, a.k.a. spectral density) to tackle this problem. A- DOGE is designed to fulfill a long desiderata of desirable characteristics. Most notably, it capitalizes on efficient approximation algorithms for DOS, that we extend to blend in node labels and attributes for the first time, making it fast and scalable for large attributed graphs and graph databases. Being based on the entire eigenspectrum of a graph, A- DOGE can capture structural and attribute properties at multiple ("glocal") scales. Moreover, it is unsupervised (i.e., agnostic to any specific objective) and lends itself to various interpretations, which makes it suitable for exploratory graph mining tasks. Finally, it processes each graph independent of others, making it amenable for streaming settings as well as parallelization. Through extensive experiments, we show the efficacy and efficiency of A- DOGE on exploratory graph analysis and graph classification tasks, where it significantly outperforms unsupervised baselines and achieves competitive performance with modern supervised GNNs, while achieving the best trade-off between accuracy and runtime.

Keywords Attributed graphs · Spectral embedding · Graph filters · Band-pass · Density of states

1 Introduction

Graphs are widely used to model structured data from different domains such as chemistry [1], biology [2], cybersecurity [3], finance [4]. The effectiveness and popularity of data-driven machine learning algorithms has necessitated expressive vector representations of

✉ Lingxiao Zhao
lingxiao@cmu.edu

Saurabh Sawlani
saurabh.sawlani@gmail.com

Leman Akoglu
lakoglu@andrew.cmu.edu

1 Heinz College, Carnegie Mellon University, Pittsburgh, PA, USA

2 SoundHound Inc., Berlin, Germany

different kinds of complex data, and graphs are no exception. Different from images or text, graphs pose novel challenges in finding effective representations as graph databases may contain graphs that vary in size and structure, and do not necessarily exhibit alignment (i.e., correspondence) between the nodes of different graphs.

Formally, we want to design a function $R : G \mapsto \mathbf{z}_G \in \mathbb{R}^D$, where D is a fixed embedding size that does not depend on the input graph size. Ideally, given a graph database with N graphs (with n nodes and m edges per graph on average), we want R to be (i) *permutation and size invariant*, where graphs with similar structure and label/attribute distribution have similar embeddings irrespective of node ordering and number of nodes, (ii) *flexible*; that leverages information from node labels and/or multi-attributes as well as edge weights, (iii) *multi-scale/glocal*; that can capture local/microscopic, mesoscopic, as well as global/macroscopic properties of a graph, and (iv) *task-agnostic/unsupervised*; that can produce embeddings independent of any downstream task or related class labels, where not being tied to a specific task allows embeddings to be general-purpose for use, e.g., in graph mining and exploratory data analysis. In addition, as with any algorithm, we want R to be (v) *efficient* and *scalable* to large graphs (large n, m) as well as large databases (large N). Finally, R that can produce one embedding at a time (vi) *independently per graph* (as opposed to "collective processing") may be desirable, which allows on-the-fly embedding per incoming graph in streaming settings, as well as embarrassing parallelization for speed.

Spectrally designed embeddings are a popular class of techniques based on the graph eigenspectrum [5], as it captures key structural graph properties, such as cuts [6], random walk stationarity [7], dynamical processes and epidemic thresholds [8], diameter, connectedness, clustering [9]. However, the high complexity of computing the eigenspectrum exactly has proven to be a barrier for creating spectrum-based graph embeddings. Moreover, while the eigenspectrum can capture important topological properties, blending in node attributes/labels into spectrally designed embeddings is non-trivial.

In this paper, we leverage fast algorithms for approximating the spectral density of a graph [10] and use it to independently construct unsupervised graph embeddings that are permutation and size invariant, flexible and multi-scale. Here, the focus is on representing the entire spectrum of the graph, which helps capture any arbitrary "band" of eigenvalues (band-pass), rather than only the extremal eigenpairs (low/high pass).

1.1 Prior work

Table 1 gives a comparison with three categories of relevant prior work in the context of desired properties for a graph embedding. These existing work do not satisfy one or more of the aforementioned properties (ii)–(vi) as we discuss next.

Unsupervised explicit graph embedding (UEGE)

Several unsupervised methods construct an explicit vector representation for each graph. Among those, spectrum-based methods have gained popularity in recent years. FGSD [11] treats a graph as a collection of spectral distances between its vertices. NetLSD [12] represents a graph as a collection of heat traces of the graph at several time points. Both methods are effective at capturing local and global structural properties of a graph; however, they ignore node labels and attributes. graph2vec [13] creates Weisfeiler–Lehman (WL) subtree-based features and learns an embedding of the graph trained to predict the existence of subtrees in the graph. It admits node labels, but ignores node attributes as well as edge weights.

Table 1 A- DOGEsatisfies all properties, while prior work miss one or more of the input graph or embedding properties

	Method	Inp. graph property			Embedding			
		Node labels	Node attributes	Edge weights	Band-pass	Task-agnostic	Scalable	Indpt. per graph
UEGE	FGSD [11], NETLSD [12]			✓		✓	✓	✓
	G2VEC [13]	✓				✓	✓	
GK	WL [14], WL- OA [15]	✓				✓		
	SAGE [16], PK [17]	✓	✓	✓		✓		
	RETGK [18]			✓		✓		
	DOSGK [19]			✓	✓	✓		
GNN	GCN [20], GIN [21]	✓	✓	✓			✓	
	CHEBNET [22], CALEYNET [23]	✓	✓	✓	✓		✓	
	A- DOGE [this paper]	✓	✓	✓	✓	✓	✓	✓

Graph kernels (GK)

Due to the existence of many effective distance measures between graphs, graph kernels are a more widely studied method of graph representations [24]. While most popular kernels are effective at capturing characteristics of the graph structure, only a few, including the Propagation Kernel (PK) [17] are able to factor in edge weights, node labels and continuous node attributes (see Table 1 in [24]).

Several graph kernels which use spectral properties have been developed in recent years. RetGK [18] represents each graph as a collection of node embeddings, where the node features are the return-probabilities of random walks of varying lengths. SAGE [16] extends this idea to graphs with labeled and attributed nodes by appending each node embedding with its one-hot encoded label and/or attributes. However, both these methods do not scale well for large graphs. Moreover, computing return probabilities of random walks tends to over-represent local features near a node, and often fails to capture global properties of the graph [19]. These issues are addressed by the density of states (DOS) GK, and its point-wise (i.e., node-level) extension (PDOS),[1] which uses Chebyshev polynomials to efficiently capture global properties of random walks, and uses fast approximation techniques [10]. However, despite their efficiency, they are limited to plain graphs and do not admit node labels or attributes.

Moreover, although graph kernels have proven effective at modeling graph structure, and in some cases node labels and attributes, for many kernel methods, computing an $N \times N$-sized kernel matrix can be restrictive in terms of both time and space, which do not scale to large databases with many graphs.

Graph neural networks (GNNs)

While most existing unsupervised embedding and kernel methods are ill-equipped to handle continuous node attributes, GNNs are able to leverage such data to a great extent. However, deep parameterized models come with their own drawbacks. They are resource-hungry, not task-agnostic, and can be slow to train. Moreover, when viewed through a spectral lens [25], most neighborhood-aggregation based GNNs such as GCN [20] and GIN [26] can only act as low-pass or high-pass filters on a graph spectrum. Only spectrally designed GNNs such as ChebNet [22] and CaleyNet [23] can act as band-pass filters.

A perhaps subtle characteristic of graph embedding methods is *independent* versus dependent/collective processing of the graphs. By design, all GNN-based methods including graph2vec require collective processing due to end-to-end training. WL and PK, respectively, obtain the compressed labels and histogram bins based on all graphs which makes them dependent. RetGK, DOSGK, and SAGE obtain graph-level embeddings through kernelizing the set of node-level embeddings, which is of different sizes across graphs, and hence they are inherently bound to create $N \times N$ pairwise kernel values rather than an explicit/independent embedding for each graph.

[1] This is called LDOS in their paper, but we use PDOS to avoid confusion with our definition of LDOS in this paper.

1.2 Our contributions

We propose A- DOGE (Attributed DOS-based Graph Embedding), for extremely fast unsupervised embedding of attributed graphs that is permutation and size invariant, flexible, and multi-scale, which is produced independently per graph.

Our main technical contributions are as follows:

- **New graph-level embedding algorithm:** We introduce a new spectrally designed graph embedding approach, called A- DOGE, that leverages the whole (eigen)spectrum of a graph. A- DOGE capitalizes on recent algorithms that can efficiently approximate the (local) density of states (L)DOS [10], extending to attributed graphs for the first time.
- **Desired characteristics:** Thanks to efficient approximations, A- DOGE is extremely fast. It can handle node labels, continuous multi-attributes, and edge weights. Leveraging the whole spectrum, it enables variable band-pass filtering as well as features that capture multi-scale properties. Further, it processes each graph independently of others, which makes it amenable for streaming scenarios as well as parallelization.
- **Exploratory graph analysis:** A- DOGE is not tied to any specific objective, which makes it suitable for both un/supervised tasks. In fact, our embedding features lend themselves to various interpretations, related to graph signal convolution, random walks, and band-filters, which prove useful in data mining and exploratory analysis of real-world graph datasets as we show through experiments.
- **Efficacy and Efficiency:** Extensive experiments show that A- DOGE is on par with or superior to all unsupervised baselines, and competitive against modern supervised GNNs on graph classification tasks. Notably, it achieves the best runtime–accuracy trade-off. (See Fig. 1.)

Reproducibility and Resources: We share all datasets and source code at https://github.com/sawlani/A-DOGE.

2 Problem statement and preliminaries

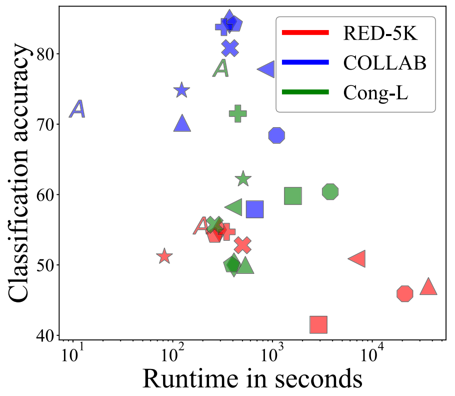
Notation. We denote scalars, vectors, matrices and sets by lowercase (x), lowercase boldface ($\mathbf{x}$), uppercase boldface ($\mathbf{X}$), and calligraphic ($\mathcal{X}$) letters, respectively. $\mathbf{X}_{:j}$ and $\mathbf{X}_{ij}$ refer to the j-th column and the (i, j)-th entry of a matrix.

We consider undirected, weighted node-attributed graphs $G = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathcal{A})$ where $\mathcal{V} = \{v_1, \ldots, v_n\}$ denotes the set of n nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of m edges. $\mathbf{W}$ depicts the weighted adjacency matrix where $\mathbf{W}_{ij} > 0$ if $(v_i, v_j) \in \mathcal{E}$, and 0 otherwise. $\mathbf{X}$ is the $n \times d$ node-attribute matrix, where $\mathcal{A} = \{a_1, \ldots, a_d\}$ denotes the set of d attributes, with $dom(a_j)$ depicting the domain of attribute a_j. In terms of graph signal processing (GSP) terminology, any $\mathbf{x} = \mathbf{X}_{:j}$ can be thought as a *graph signal* on the nodes, with one scalar per node.

Problem 1 (Unsupervised Graph-level Embedding) **Given** a set of undirected, weighted and node-attributed/labeled graphs $\mathcal{G} = \{G_1, \ldots, G_N\}$, for $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathbf{X}_i, \mathcal{A})$, where

(i) graphs in $\mathcal{G}$ can be of varying sizes, (ii) there exists no particular correspondence between the nodes of different graphs, and (iii) the (categorical and/or continuous) attributes and their domain are shared among all graphs,

Find D-dimensional graph-level embedding $\mathbf{z}_G \in \mathbb{R}^D$ for each $G \in \mathcal{G}$ that captures both structural and attribute information.



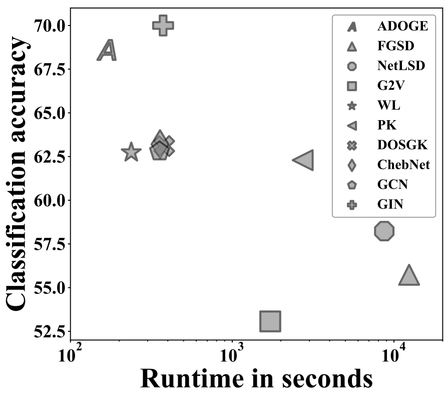


Fig. 1 A- DOGE achieves superior runtime–performance trade-off as compared to 9 graph classification baselines (See Sect. 4.1). (left) Runtime (log-scale) vs. accuracy on three individual datasets: `REDDIT-5K` with largest N, `COLLAB` with largest $avg(m)$ and the largest synthetic dataset `Congress-L` w.r.t. $avg(n)$. Colors correspond to separate datasets. (right) Average runtime vs. graph classification accuracy across datasets. Symbols depict different methods (Symbol A stands for A- DOGE)

Let $\widetilde{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ denote the symmetrically normalized adjacency matrix, where $\mathbf{D}$ is the diagonal degree matrix with $\mathbf{D}_{i,i} = \sum_j \mathbf{W}_{i,j}$. Let $\widetilde{\mathbf{L}} = I - \widetilde{\mathbf{W}}$ denote the Laplacian matrix, and $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ the random walk matrix. For a connected graph, $\widetilde{\mathbf{W}}$ has eigenvalues $-1 = \lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{n-1} = 1$ with corresponding eigenvectors $\{\mathbf{u}_k\}_{k=0}^{n-1}$. $\widetilde{\mathbf{W}}$ has the same set of eigenvectors as $\widetilde{\mathbf{L}}$ whose eigenvalues are the shifted set $\{\mu_k = 1 - \lambda_k\}_{k=0}^{n-1} \in [0, 2]$. $\widetilde{\mathbf{W}}$ also shares the same eigenvalues as $\mathbf{P}$. As such, the spectral density function has bounded support for these graph matrices. Following GSP convention, we refer to the eigenvalues as the **graph frequencies**.

In this work, we use $\widetilde{\mathbf{W}}$ as the so-called graph shift operator $\mathbf{S}$ which generalizes to any symmetric matrix of a graph. Let $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ depict the eigendecomposition, where $\mathbf{\Lambda} := \text{diag}([\lambda_1 \ldots \lambda_n])$ and $\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_n]$.

Definition 1 (Graph spectrum) The **spectrum** of a graph is composed of the set of the graph eigenvalues, together with their multiplicities, of the (normalized) adjacency matrix.

Graph Fourier transform. The graph Fourier transform (GFT) of a graph signal $\mathbf{x} \in \mathbb{R}^n$ is defined as the projection

$$\widehat{\mathbf{y}} = \mathcal{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$$

and the inverse GFT of $\widehat{\mathbf{y}} \in \mathbb{R}^n$ is given as

$$\mathbf{x} = \mathcal{F}^{-1}(\widehat{\mathbf{y}}) = \mathbf{U}\widehat{\mathbf{y}}$$

Graph filtering. A graph filter is an operation on a graph signal with output in the graph frequency domain, that is,

$$\widehat{\mathbf{y}}_{flt} = \phi(\mathbf{\Lambda})\widehat{\mathbf{y}}\ , \tag{1}$$

where $\phi(\mathbf{\Lambda})$ is a diagonal matrix with filter frequency response values as its diagonal elements.

Definition 2 (Frequency Response Function (FRF)) The **frequency response function** of a graph filter is written as

$$\phi : \mathbb{C} \mapsto \mathbb{R}, \quad \lambda_i \to \phi(\lambda_i)\ , \tag{2}$$

which, simply put, assigns a scalar value $\phi(\lambda_i)$ to each graph frequency (i.e., eigenvalue) λ_i.

By applying the inverse GFT on both sides of Eq. (1), we can get the filter output in the node domain as

$$\mathbf{x}_{flt} = \mathbf{U}\phi(\mathbf{\Lambda})\widehat{\mathbf{y}} = \mathbf{U}\phi(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x} = \phi(\mathbf{S})\mathbf{x} \ .$$

Signal convolution. Graph convolution of two signals, say $\mathbf{x}$ and $\mathbf{x}'$, each in $\mathbb{R}^n$, yields another signal $\mathbf{c} \in \mathbb{R}^n$ as

$$\mathbf{c}_{\mathbf{x},\mathbf{x}'} = \mathbf{x} *_G \mathbf{x}' = \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{x}') = \sum_{i=1}^{n} \mathbf{u}_i(\mathbf{u}_i^T\mathbf{x})(\mathbf{u}_i^T\mathbf{x}')$$

where $\odot$ depicts the Hadamard product. We can write the Fourier transform of the convolution as

$$\mathcal{F}(\mathbf{c}_{\mathbf{x},\mathbf{x}'}) = \widehat{\mathbf{c}}_{\mathbf{x},\mathbf{x}'} = \{(\mathbf{u}_i^T\mathbf{x})(\mathbf{u}_i^T\mathbf{x}')\}_{i=1}^{n} \ . \tag{3}$$

Density of States. Spectral density is the overall distribution of the eigenvalues as induced by any symmetric $n \times n$ graph matrix $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. It is also referred to as the density of states (DOS) in the physics literature, reflecting the number of states at different energy levels [27]. Formally,

Definition 3 (Density of States (DOS)) DOS or the spectral density induced by $\mathbf{S}$ is the density function

$$f(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \delta(\lambda - \lambda_i) \ , \tag{4}$$

where $\delta(\cdot)$ is the Dirac delta function.

Definition 4 (Local Density of States (LDOS)) Likewise, for *any* input vector $\mathbf{v} \in \mathbb{R}^n$, LDOS is given as

$$f(\lambda; \mathbf{v}) = \sum_{i=1}^{n} (\mathbf{v}^T\mathbf{u}_i)^2 \delta(\lambda - \lambda_i) \ . \tag{5}$$

The following related equalities can be derived easily, respectively, for DOS and LDOS.

$$f(\lambda) = \frac{1}{n} f(\lambda; \sum_{i=1}^{n} \mathbf{u}_i) \tag{6}$$

$$f(\lambda) = \frac{1}{n} E_{\mathbf{z} \sim N(0,1)[f(\lambda;\mathbf{z})]} \tag{7}$$

$$\int \phi(\lambda) f(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \phi(\lambda_i) = \frac{\text{trace}(\phi(\mathbf{S}))}{n} \tag{8}$$

$$\int \phi(\lambda) f(\lambda; \mathbf{v}) = \sum_{i=1}^{n} \phi(\lambda_i)(\mathbf{v}^T\mathbf{u}_i)(\mathbf{u}_i^T\mathbf{v}) = \mathbf{v}^T\phi(\mathbf{S})\mathbf{v} \tag{9}$$

Scaling (L)DOS. The extremal (i.e., a few top largest or smallest) eigenpairs of various graph matrices have been associated with important graph characteristics, such as small-cut partitions [6], convergence rate of random walks to stationarity [7], unfolding of dynamical processes and epidemic thresholds [8], etc. Obtaining those few eigenpairs is also computationally easy. On the other hand, (L)DOS provides the distribution of the *entire* spectrum, which opens the door for the analysis of graph properties that are not evident from only the extremal eigenpairs. However, computing all n eigenvalues and eigenvectors of a graph with n nodes is considerably more demanding. Therefore, analyzing large graphs through their density of states has been obstructed by the lack of scalable algorithms, until recently.

In their award-winning work, Dong *et al.* [10] introduced highly efficient approximation algorithms to compute spectral densities, scalable to graphs with as large as tens of millions of nodes and billions of edges. Their main focus has been scaling the computation of these functions, with approximation-error analysis on plain graphs. In this paper, we capitalize on their work for speed and extend it to leverage node attributes for the first time toward fast, *attributed* graph-level embedding.

Fast Approximation of (L)DOS. As introduced in [10], there are two methods for estimating spectral densities: the kernel polynomial method (KPM) and the Gauss quadrature via Lanczos iteration (GQL). KPM expands the (L)DOS with orthogonal polynomial base functions, and the typical polynomial basis is the Chebyshev polynomials. Chebyshev approximation requires the eigenvalues of input matrix to be supported on $[-1, 1]$, which is satisfied by the graph shift operator $\mathbf{S}$. In practice, only a finite number of moments are needed to approximate the (L)DOS well, especially for smooth (L)DOS. Dong et al. [10] also proposed some pre-conditioning step to accelerate the error decay with respect to the number of moments. In this paper we use GQL to estimate (L)DOS which we introduce in detail as follows.

Gauss quadrature (GQ) is a numerical method to estimate definite integral of a function with a weighted sum of function values at specified points, and it has been applied to computing $\mathbf{u}^T g(\mathbf{A})\mathbf{u}$ for an arbitrary vector $\mathbf{u}$, a symmetric positive-definite $n \times n$ matrix $\mathbf{A}$ and a matrix function $g(\cdot)$. Note that $\mathbf{u}^T g(\mathbf{A})\mathbf{u}$ can be re-written as Riemann–Stieltjes integral [28]; letting $\mathbf{A} = Q^T \Lambda Q$ upon eigen-decomposition and $\tilde{\mathbf{u}} = Q\mathbf{u}$:

$$\mathbf{u}^T g(\mathbf{A})\mathbf{u} = \tilde{\mathbf{u}}^T g(\Lambda)\tilde{\mathbf{u}} = \sum_{i=1}^{n} g(\lambda_i)\tilde{\mathbf{u}}_i^2 = \int_{\lambda_{\min}}^{\lambda_{\max}} g(\lambda)\mathrm{d}\alpha(\lambda) \tag{10}$$

where $\alpha(\lambda)$ is a piece-wise constant function defined as

$$\alpha(\lambda) = \begin{cases} 0 & \lambda < \lambda_1 \\ \sum_{i=1}^{k} \tilde{\mathbf{u}}_i^2 & \lambda_k \leq \lambda < \lambda_{k+1},\ k = 1, \ldots, n-1 \\ \sum_{i=1}^{n} \tilde{\mathbf{u}}_i^2 & \lambda \geq \lambda_n \end{cases} \tag{11}$$

Using GQ, we can approximate the integral as $\mathbf{u}^T g(A)\mathbf{u} = \int_{\lambda_{\min}}^{\lambda_{\max}} g(\lambda)\mathrm{d}\alpha(\lambda) \approx \sum_{i=1}^{p} w_i g(\theta_i)$ with some weights $\{w_i\}_{i=1}^{p}$ and points $\{\theta_i\}_{i=1}^{p}$. Different ways of computing $\{w_i\}_{i=1}^{p}$ and $\{\theta_i\}_{i=1}^{p}$ have been summarized in [28], and one way is called Gauss quadrature via Lanczos iteration (GQL).

Before introducing GQL, let us build the connection between (L)DOS and computing $\mathbf{u}^T g(\mathbf{A})\mathbf{u}$. Expanding the definition of LDOS in Eq. (5), we can write

$$f(\lambda; \mathbf{v}) = \sum_{i=1}^{n} (\mathbf{v}^T \mathbf{u}_i)^2 \delta(\lambda - \lambda_i) = (\mathbf{U}\mathbf{v})^T \delta(\lambda - \mathbf{\Lambda})(\mathbf{U}\mathbf{v}) = \mathbf{v}^T \delta(\lambda - \mathbf{S})\mathbf{v}\ . \tag{12}$$

The above formulation indicates that LDOS can be represented in the form of $\mathbf{u}^T g(\mathbf{A})\mathbf{u}$ by substituting $\mathbf{u} \leftarrow \mathbf{v}$, $\mathbf{A} \leftarrow \mathbf{S}$, and $g(x) \leftarrow \delta(\lambda - x)$, and hence can be approximated via GQL.

To generate $\{w_i\}_{i=1}^{p}$ and $\{\theta_i\}_{i=1}^{p}$ for GQ approximation of $\mathbf{v}^T \delta(\lambda - \mathbf{S})\mathbf{v}$, Lanczos algorithm is first conducted to decompose $\mathbf{S}$ into a tridiagonal matrix $\mathbf{T}_k \in \mathbb{R}^{k \times k}$ with k being the number of iterations of Lanczos algorithm. Given a matrix $\mathbf{S}$ and an initial vector $\mathbf{z}_1$ with $\|\mathbf{z}_1\|^2 = 1$, Lanczos algorithm iteratively generates k orthogonal unit vectors $\mathbf{z}_1, \ldots, \mathbf{z}_k$ and outputs a tridiagonal matrix $\mathbf{T}_k$. Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_k] \in \mathbb{R}^{n\times k}$, $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_k$, then

$$\mathbf{S}\mathbf{Z} = \mathbf{Z}\mathbf{T}_k + \mathbf{R} \tag{13}$$

where $\mathbf{R} \in \mathbb{R}^{n\times k}$ is the residual term and each column vector of $\mathbf{R}$ is orthogonal to $\mathbf{z}_i, \forall i$. Hence $\mathbf{Z}^T\mathbf{R} = \mathbf{0}_k$.

A well-known theorem characterizes the relationship between Lanczos algorithm and GQ approximation, as stated below.

Theorem 1 ([29, 30]) *The eigenvalues of T_k form the points $\{\theta_i\}_{i=1}^k$ of Gauss quadrature, and the weights $\{w_i\}_{i=1}^k$ are given by the squares of the first elements of the eigenvectors of T_k.*

The eigendecomposition for tridiagnoal matrix T_k is fast with complexity $O(k^2)$. Let $\{(\tau_1, \mathbf{c}_i), \ldots, (\tau_k, \mathbf{c}_k)\}$ be the eigenvalues and eigenvectors of T_k. Theorem 1 implies that $\mathbf{z}_1^T g(\mathbf{S})\mathbf{z}_1 \approx \sum_{i=1}^k (\mathbf{e}_1^T \mathbf{c}_i)^2\, g(\tau_i) = \sum_{i=1}^k \mathbf{c}_{i1}^2\, g(\tau_i)$. Replacing the starting vector $\mathbf{z}_1$ of Lanczos to $\frac{\mathbf{v}}{\|\mathbf{v}\|}$ and $g(x)$ to $\delta(\lambda - x)$ we can get the approximation of LDOS as follows

$$f(\lambda; \mathbf{v}) = \mathbf{v}^T\delta(\lambda - \mathbf{S})\mathbf{v} \approx \|\mathbf{v}\|^2 \sum_{i=1}^k \mathbf{c}_{i1}^2 \delta(\lambda - \tau_i) \tag{14}$$

Following Eq. (7), DOS can be computed as

$$f(\lambda) = \frac{1}{n} E_{\mathbf{z}\sim N(0,1)}[f(\lambda; \mathbf{z})] = \frac{1}{n \times p} \sum_{i=1}^p f(\lambda; \mathbf{z}_i) \tag{15}$$

where $\mathbf{z}_1, \ldots, \mathbf{z}_p$ are p random vectors from normal distribution.

3 Graph-level embedding with A- DOGE

3.1 Motivation

Our spectrally designed A- DOGE derives graph-level features based on the node attributes and the *entire* spectrum of $\widetilde{\mathbf{W}}$ (can be other symmetric graph matrix, w.l.o.g. referred as $\mathbf{S}$, see Sect. 2), where the spectrum is composed of *all* of the eigenvalues. Before delving into details, we discuss the motive for using the full spectrum and present an illustrative example.

Why the entire spectrum? We design graph-level features based on all of the eigenpairs of a graph matrix for two primary reasons. First, a large number of studies have found that the full eigenvalue spectra of different classes of real-world networks differ considerably [9, 31–33]. This suggests that the spectra can play a key discriminative role. Second, real-world networks are observed to exhibit localization on low-order eigenvectors, which are those eigenvectors associated with the non-extremal eigenvalues (in the sense of being the largest or smallest), but that are "buried" further down in the eigenvalue spectrum [34]. Notably, they capture *mesoscopic inhomogeneity* in networks which is defined as topologically distinct groupings of nodes, from few nodes to large modules, communities, or different interconnected subnetworks [35].

Illustrative example: To illustrate the valuable information that non-extremal eigenpairs carry, we present a visual analysis of low-order eigenvector localization using the `MIG` graph

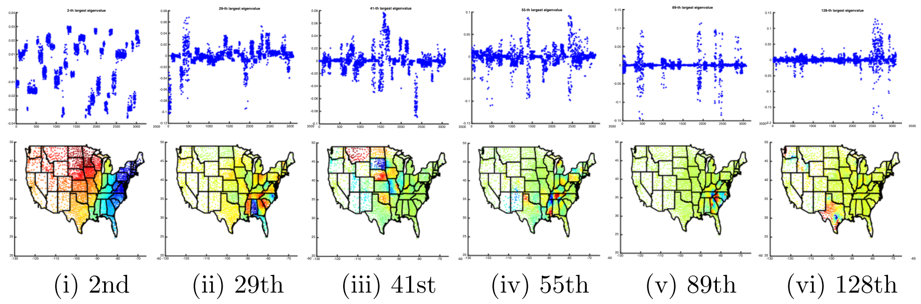


Fig. 2 (top) Eigenvector entries (y-axis) versus node (i.e., US county) index (x-axis) for (from left to right) the top 2nd, 29th, 41st, 55th, 89th, and 128th eigenvector of the `MIG` graph (See Sect. 4.1); (bottom) Eigenvector entries as heatmap (red: high to blue: low) for nodes (US counties) shown in 2-d coordinates, solid black lines depict official US state borders (best in color)

(See Sect. 4.1). It consists of the counties across 49 mainland US states as nodes, and an edge depicts the total number of people that migrated between two counties during 1995–2000 [34].

Eigenvector localization arises when most of the entries of an eigenvector are zero or near-zero and implies that the nonzero components of the eigenvector coincide with a particular set of geometrically distinguished nodes in the graph. Extremal eigenvectors typically exhibit low localization; as shown in Fig. 2(i), the 2nd eigenvector has many nonzeros and mainly captures macroscopic properties, in this case, the graph cut depicting relatively fewer migrations between west- and east-coasts. Lower-order eigenvectors, as shown in (ii)–(iv), reflect mesoscopic structure in terms of migration patterns. For example, the 41st eigenvector depicts migration in and around South Dakota. On the other hand, even lower eigenvectors localize increasingly, narrowing in a few counties, as shown in (v) and (vi). For example, the 128th eigenvector has localized to a few counties within Texas near Austin, reflecting microscopic patterns. It is remarkable that the low-order eigenvectors align with geographical and political boundaries, carrying useful information at multiple scales.

3.2 Spectrum as histogram: DOS, LDOS, cLDOS features

3.2.1 Density of states (DOS)

DOS or spectral density as given in Eq. (4) is a continuous probability density function $f(\lambda)$ of the eigenvalues. We represent it with a histogram density estimator, denoted $h^{DOS}(\lambda)$ that partitions the eigenvalue range $[-1, 1]$ for $\widetilde{\mathbf{W}}$ into $B = 2/\Delta$ disjoint bins of equal width Δ. Let us denote their centers by $\widetilde{\lambda}_b$ for $b \in \{1, \ldots, B\}$. For any $i \in \{1, \ldots, n\}$, let $\text{Bin}(\lambda_i)$ denote the bin that λ_i belongs to. We define our DOS histogram features for a graph as follows.

Definition 5 (DOS histogram features) DOS histogram is a B-dimensional vector, denoted $\mathbf{h}^{\text{DOS}} \in \mathbb{R}^B$, where

$$\mathbf{h}^{\text{DOS}}(\widetilde{\lambda}_b) = \frac{1}{\Delta}\frac{\sum_{i=1}^{n} \mathbb{I}(\lambda_i \in Bin(\widetilde{\lambda}_b))}{n}, b \in \{1, \ldots, B\} \tag{16}$$

3.2.2 Local density of states (LDOS)

We also represent the *local* density of states (LDOS) in Eq. (5) similarly and define LDOS histogram features.

Definition 6 (LDOS histogram features) For a given vector $\mathbf{v} \in \mathbb{R}^n$, the LDOS histogram is a B-dimensional vector, denoted $\mathbf{h}_{\mathbf{v}}^{\text{LDOS}} \in \mathbb{R}^B$, where

$$\mathbf{h}_{\mathbf{v}}^{\text{LDOS}}(\widetilde{\lambda}_b) = \frac{1}{\Delta} \frac{\sum_{i=1}^{n} (\mathbf{v}^T \mathbf{u}_i)^2 \, \mathbb{I}(\lambda_i \in Bin(\widetilde{\lambda}_b))}{n} \text{ , } \forall b \tag{17}$$

Note that by abusing convention slightly, we use the word histogram to refer to Eq. (17) although it is not a normalized density mass function. Figure 3 shows examples to DOS (top) and LDOS (middle & bottom) histograms with $B = 40$ each.

Computing both DOS and LDOS histograms requires all of the eigenvalues $\lambda_i, i = \{1 \ldots n\}$ for a graph with n nodes. Further, LDOS requires all the corresponding eigenvectors $\mathbf{u}_i$'s. For even moderate size graphs, computing the complete set of eigenpairs is prohibitive. Most recently, Dong et al. [10] introduced fast and scalable approximation algorithms to estimate these spectral densities. Our work is inspired by and builds on their work to efficiently obtain both $\mathbf{h}^{\text{DOS}}$ and $\mathbf{h}^{\text{LDOS}}$ based on the Gauss Quadrature and Lanczos (GQL) algorithm [36].

On the other hand, both in [10] and their follow-up work [19], $\mathbf{v} = \mathbf{e}_i$ is used in Eq. (5) to capture the spectral information about each particular node $i = \{1, \ldots, n\}$, called point-wise density of states (PDOS), where $\mathbf{e}_i$ is the i-th standard basis vector with i-th entry equal to 1 and 0 elsewhere. As such, both works are limited to plain graphs without node labels/attributes. We extend the use of LDOS to attributed graphs for the first time, *by setting* $\mathbf{v} \in \mathbb{R}^n$ in Eq. (17) to *capture a graph signal on the nodes associated with an attribute*.

Specifically, given a categorical or binary attribute a_j, we create a separate $\mathbf{v}$ for each unique value $val \in dom(a_j)$ where $\mathbf{v}_i := 1$ if $\mathbf{X}_{ij} = val$ and 0 otherwise. For numerical attributes, we set $\mathbf{v} := \overline{\mathbf{X}}_{:j}$ where $\overline{\mathbf{X}}$ denotes the column-wise standardized attribute matrix. Notably, LDOS can be extended to structural node-level attributes, such as degree or other node centrality measures and eccentricity.

Interpreting LDOS. There is an intuitive interpretation of a LDOS feature in Eq. (17). The term $\mathbf{v}^T \mathbf{u}_i$, that is the dot product between an attribute vector and a graph eigenvector, is to reflect the *alignment* between attribute values and the structurally distinct group of nodes that the eigenvector captures. The better the alignment, the larger is the LDOS feature value for the bin that the corresponding eigenvalue falls into.

Why the attribute-based LDOS? We provide an illustrative example, motivating the use of LDOS besides DOS-based histogram features. To this end, we use our `Congress` graph, as described in experiments Sect. 4.1. It consists of US senators as nodes across 41 US Senates from the 70th to 110th Congress, where weighted edges capture voting agreement. Each node is labeled with party affiliation; as Democrat, Republican, or Other. Figure 4a gives the spy plot for the adjacency matrix, where the dense blocks on the diagonal correspond to each one of 41 Senates. Cross-senate edges connect the same senator who appear across multiple senates.

From the `Congress` graph, we create two variants. We first select only one Senate at random. Next, we add noise edges between the same-party nodes in the first variant, called `Congress`-within, and among random nodes in the second variant, called `Congress`-rand. As such, the structural difference between the variants is associated with node labels. The edge weights are chosen uniformly at random from [0.25, 1]. The total weight of edges

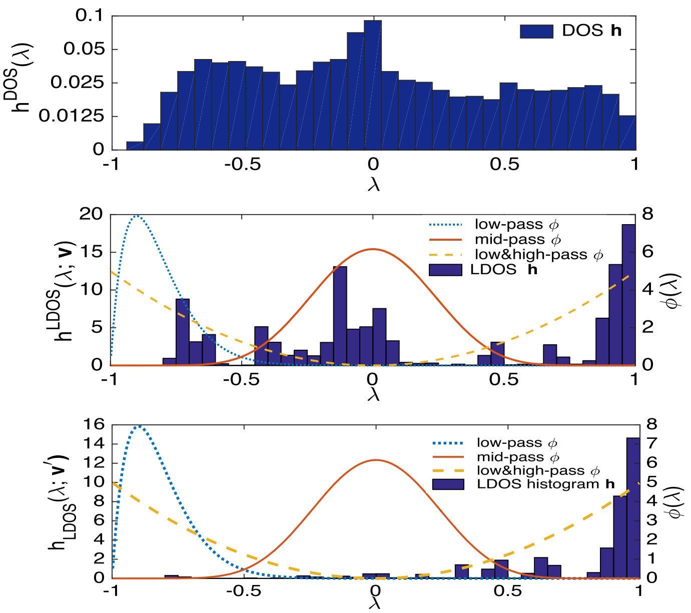
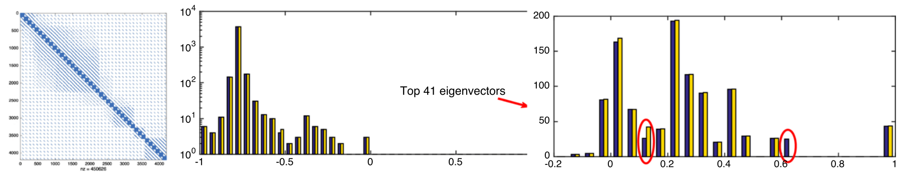


Fig. 3 Example (top) histogram density estimator of the spectral density, a.k.a. density of states (DOS), of a graph for symmetrically normalized $\tilde{\mathbf{W}}$ with eigenvalues (i.e., frequencies) in $[-1, 1]$, and local density of states (LDOS) histogram for two different vectors (middle) $\mathbf{v}$ and (bottom) $\mathbf{v}'$. Also shown (in color) on LDOS plots are three different frequency response functions over the spectrum; low-pass (blue dotted), both low- and high-pass (orange dashed), and mid-pass (red solid)



(a) `Congress` (b) DOS histograms (c) LDOS histograms, for Democrat

Fig. 4 From the `Congress` graph in (**a**) we create two variants: to `Congress`-within (blue) we add noise edges among same-party nodes from one (out of 41) Senate only, whereas to `Congress`-rand (orange) we add noise edges among randomly chosen nodes from the same Senate. As topology is perturbed only slightly, their DOS histograms in (**b**) are very similar, while LDOS histogram features in (**c**) reflect key differences, as highlighted in red ellipses

added to each graph is exactly the same. As we only perturb one of 41 senates in this way, the two variants share mostly the same topology. As a result, their DOS histograms are hard to distinguish, as shown in Fig. 4b, where the right-most bump depicts the top 41 eigenvectors (each close to 1) capturing the 41 dense subgraphs per Senate. In contrast, LDOS histograms (using party affiliation Democrat, Republican is similar) reflect clear differences as shown in Fig. 4c.

3.2.3 Coupled local density of states (cLDOS)

In addition to the original LDOS, we also create *interaction* features between pairs of node attributes. Accordingly, the coupled-LDOS histogram features are defined as follows.

Definition 7 (cLDOS histogram features) For two input vectors $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^n$, the coupled-LDOS histogram is a B-dimensional vector, denoted $\mathbf{h}^{\text{cLDOS}}_{\mathbf{v},\mathbf{v}'} \in \mathbb{R}^B$, where

$$\mathbf{h}^{\text{cLDOS}}_{\mathbf{v},\mathbf{v}'}(\widetilde{\lambda}_b) = \frac{1}{\Delta} \frac{\sum_{i=1}^{n} (\mathbf{v}^T \mathbf{u}_i)(\mathbf{u}_i^T \mathbf{v}') \ \mathbb{I}(\lambda_i \in Bin(\widetilde{\lambda}_b))}{n}, \forall b \tag{18}$$

Note that $(\mathbf{v}^T \mathbf{u}_i)(\mathbf{u}_i^T \mathbf{v}')$ is the i-th entry of $\widehat{\mathbf{c}}_{\mathbf{v},\mathbf{v}'}$ (from Eq. 3). Hence $\mathbf{h}^{\text{cLDOS}}_{\mathbf{v},\mathbf{v}'}$ can simply be viewed as $\widehat{\mathbf{c}}_{\mathbf{v},\mathbf{v}'}$, binned according to the corresponding eigenvalues of each entry.

Moreover, recall that we use the GQL algorithm to approximate the LDOS features, where the terms $\mathbf{v}^T \mathbf{u}_i$ or $\mathbf{u}_i^T \mathbf{v}'$ are not computed using the individual eigenvectors explicitly. Nevertheless it is easy to acquire cLDOS features in Eq. (18) using the LDOS features in Eq. 17 and simple algebra. Given the separate LDOS features for $\mathbf{v}$ and $\mathbf{v}'$, we also create those for $(\mathbf{v} + \mathbf{v}')$. Then,

$$\mathbf{h}^{\text{cLDOS}}_{\mathbf{v},\mathbf{v}'} = \left[\mathbf{h}^{\text{LDOS}}_{\mathbf{v}+\mathbf{v}'} - \mathbf{h}^{\text{LDOS}}_{\mathbf{v}} - \mathbf{h}^{\text{LDOS}}_{\mathbf{v}'}\right]/2. \tag{19}$$

3.3 Functions over the spectrum: aggregate features

DOS, LDOS and cLDOS histograms provide “raw” information about the graph spectrum and the attributes. In addition, we define aggregate scalar features by specifying various frequency response functions (FRF) [25] (Eq. (2)) over these histograms.

Definition 8 ((cL)DOS aggregate features) Given a DOS, LDOS or cLDOS histogram $\mathbf{h} \in \mathbb{R}^B$, and a frequency response func. $\phi(\cdot)$, a (cL)DOS aggregate feature $g_\phi \in \mathbb{R}$ is written as

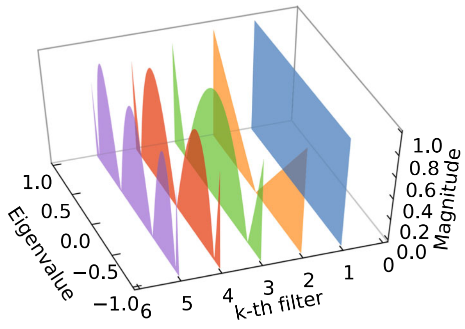
$$g_\phi = \sum_{b=1}^{B} \mathbf{h}(\widetilde{\lambda}_b) \ \phi(\widetilde{\lambda}_b) \tag{20}$$

Each FRF $\phi(\cdot)$ focuses on a different part of the spectrum, inducing a variety of graph filters. In Fig. 3 (bottom), we show three example FRFs; a low-pass one (blue) that has high values for smaller eigenvalues, a mid-pass one (red) as well as one that is both low-and-high pass (orange). To extract graph connectivity and attribute information broadly, we construct a *portfolio* of these graph filters, i.e., associated FRF’s $\{\phi(\cdot)\}$, called a **filterbank**.

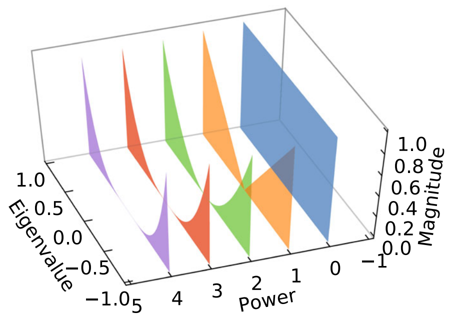
Before delving into the details of our filterbank, we make a few remarks. First, note that the sum in Eq. (20) is an approximation of the integral in Eq. (8) for $\mathbf{h}^{\text{DOS}}$, that of Eq. (9) for $\mathbf{h}^{\text{LDOS}}$, and accordingly an approximation of $\mathbf{v}^T \phi(\mathbf{S})\mathbf{v}'$ for $\mathbf{h}^{\text{cLDOS}}$. Second, given the efficiently computed histograms thanks to the GQL algorithm, computing the aggregate features by Eq. (20) is extremely fast and simply involves a weighted sum. This allows us to employ a large filterbank containing many different FRF’s almost for “free”. Finally, we have seen that our cLDOS aggregate features relate to graph signal convolution. Denoting the vector of frequency responses by $\boldsymbol{\phi} := \{\phi(\lambda_i)\}_{i=1}^n$, based on Eqs. (9) and (3),

$$\int \phi(\lambda) f(\lambda; \mathbf{v}, \mathbf{v}') = \sum_{i=1}^{n} \phi(\lambda_i)(\mathbf{v}^T \mathbf{u}_i)(\mathbf{u}_i^T \mathbf{v}) = \boldsymbol{\phi}^T \widehat{\mathbf{c}}_{\mathbf{v},\mathbf{v}'} \tag{21}$$

In the following, we present two classes of FRF’s that A- DOGE employs to extract (cL)DOS aggregate features.



(a) Chebyshev polynomials



(b) Power functions

Fig. 5 Frequency response functions $\phi_k(\lambda)$. Note that magnitude is in absolute values

3.3.1 Chebyshev polynomials

We use the series of Chebyshev polynomials as a set of FRF's defined by the recurrence $\phi_1(\lambda) = 1,\ \phi_2(\lambda) = 2(\lambda/\lambda_{\max}) - 1$, and $\phi_k(\lambda) = 2\phi_2(\lambda)\phi_{k-1}(\lambda) - \phi_{k-2}(\lambda)$, where $\lambda_{\max}$ is the maximum eigenvalue.

Interpretation. As shown in Fig. 5a, Chebyshev polynomials provide frequency profiles that cover various parts of the spectrum. For example, the 2nd one is mostly a low- and high-pass filter and stops the middle band, while the 3rd one passes the middle bands as well as very high and very low bands of the spectrum, and so on. Given a number of these FRF's, emphasis can be put on passing/stopping specific bands by a weighted combination of them.

The flexibility of any-band filtering by A- DOGE is favorable over several modern graph neural networks (GNNs). GCN [20], for instance, works as a low-pass-only filter and hence does not cover the whole spectrum. GIN [21] has a learnable scalar parameter ϵ that determines which band to stop, however its FRF is a linearly decreasing function, which is not a strong low-pass or high-pass filter. (See Fig. 2 in [25].) In contrast, spectrally designed ChebNet [22] is more expressive and also employs the Chebyshev polynomials. We compare to these modern GNNs in the experiments on graph classification tasks.

3.3.2 Power functions

The second class of FRF's in our filterbank uses (both positive and negative) powers of the spectrum, that is,

$$\phi_k(\lambda) = \lambda^k\ ,\ k = \pm\{1, \ldots, K/2\}$$

Interpretation. Our aggregate features using the power functions relate to random walks on the graph. Consider positive values of k and $\mathbf{S} = \widetilde{\mathbf{W}}$. Recall that for $\mathbf{h}^{\text{DOS}}$, Eq. (20) is an approximation of trace($\phi(\mathbf{S})$), which is equal to the total return-probability of a k-step random walk to a node. For $\mathbf{h}^{\text{LDOS}}$, aggregate features approximate $\mathbf{v}^T\phi(\mathbf{S})\mathbf{v}$. For a binary/categorical attribute where $\mathbf{v}$ depicts a certain value, e.g., $val :=$ (`party_affiliation:democrat`), it corresponds to the probability that a k-step random walk starting at any node with value val "hits"/reaches another node with the same value. For $\mathbf{h}^{\text{cLDOS}}$, similarly, it is the probability that such a walk starting at any node with a certain val will reach another node with a different val'. Moreover, for two continuous attributes $\mathbf{v}$ and $\mathbf{v}'$, approximating $\mathbf{v}^T\phi(\mathbf{S})\mathbf{v}'$ via $\mathbf{h}^{\text{cLDOS}}$ would capture the covariance of the attributes over "k-hop connected" pairs of nodes that can reach each other within k-steps.

Table 2 Summary of A- DOGE graph-level features based on spectral densities (cL)DOS, organized as (a) histogram features, resp. for DOS (Definition 5), LDOS (Definition 6), cLDOS (Definition 7), and (b) aggregate features (Definition 8) using (b.i) Chebyshev polynomials and (b.ii) power functions

DOS			LDOS			cLDOS		
hist	agg. g_ϕ		**h**ist	agg. g_ϕ		**h**ist	agg. g_ϕ	
	Cheb	Pow		Cheb	Pow		Cheb	Pow
B	K	K	BD	KD	KD	$B\binom{D}{2}$	$K\binom{D}{2}$	$K\binom{D}{2}$

B: # of histogram bins, K: # of FRF's, D: # of node attributes

The interpretations extend to the negative powers as well, which correspond to many walks of different lengths in the limit. In that respect, aggregate features using power functions depict *multi-scale* properties, where increasingly positive values of k capture microscopic to mesoscopic properties related to short/local random walks, whereas negative powers relate to the long-range walks and thereby macroscopic structure.

3.4 A- DOGE: overall summary

We conclude with an overview of all the graph-level features described in this section. Table 2 gives the number of features by category, where B is the number of histogram bins, K is the number of Chebyshev or power frequency response functions (FRF's), and $D \geq d$ is the total number of attributes upon one-hot-encoding the categorical and binary attributes. A- DOGE yields $(B+2K)(1+D+\binom{D}{2})$ features in total for an attributed graph, which are permutation- and size-invariant, task-agnostic, variable band-pass, multi-scale, and extremely efficient to compute. We outline the steps in Fig. 6 and give detailed complexity analysis next.

Extension to more features. We note that one may expand the set of A- DOGE features in two possible ways. First, the input vector(s) to (c)LDOS, denoted $\mathbf{v}$ (and $\mathbf{v}'$) in Definition 6 (and in Definition 7), are flexible. Besides a vector depicting one node attribute at a time, one can design features of features or even include other topological properties of the nodes. Second, one can define FRF's of other forms, aiming to capture different functions of the spectrum. Those could include very flexible yet perhaps less interpretable functions, especially if the downstream task is more performance-oriented and less human-centered.

Extension to directed graphs. The original method is designed based on eigendecomposition of symmetric matrix, and the fast computation of DOS and LDOS also relies on symmetric matrix. For directed graph, directly using the eigenvalues and eigenvectors of the directed adjacency matrix introduces complex numbers that are hard to define its distributions, and the fast algorithm of computing eigenvectors is not available yet. To avoid the problem and still extend the designed method to directed graphs, one can apply the proposed method over a transformed undirected graph of the directed graph, instead of working with its directed adjacency matrix. Specifically, one can transform a directed graph to a undirected graph reversibly, by replacing each directed edge $e = (x, y)$ with 5 new vertices $v_1^e, \ldots, v_5^e$ and new edges (x, v_1^e), (v_3^e, y), (v_1^e, v_3^e), (v_3^e, v_4^e), (v_4^e, v_5^e). Notice that the generated undirected graph can be transform back to original directed graph by identifying all added new nodes with degree information.[2] With the help of the injective transformation between directed graph and undirected graph, our method can be used to encode directed graph, with introducing some computational cost.

[2] https://cs.stackexchange.com/questions/19744/converting-a-digraph-to-an-undirected-graph-in-a-reversible-way.

Input: Graph G (with D node attributes), parameters B, K.
- (Preprocess) Compute $2K$ agg. functions on B bin centers.
- $\widetilde{\mathbf{W}} \leftarrow$ normalized adjacency matrix of G
- $\mathbf{v}_d \leftarrow$ attribute vector for each $d \in \{1 \dots D\}$
- Compute $\mathbf{h}^{\mathrm{DOS}}$ using GQL [10] on $\widetilde{\mathbf{W}}$
- For each d, compute $\mathbf{h}^{\mathrm{LDOS}}_{\mathbf{v}_d}$ using GQL on $\widetilde{\mathbf{W}}, \mathbf{v}_d$.
- For each pair $d, d' \in \{1 \dots D\}$, compute $\mathbf{h}^{\mathrm{LDOS}}_{\mathbf{v}_d+\mathbf{v}_{d'}}$ using GQL and then $\mathbf{h}^{\mathrm{cLDOS}}_{\mathbf{v}_d,\mathbf{v}_{d'}}$ using Eq. (19).
- For each histogram computed above, dot product with all aggregate functions to produce aggregate features.

Fig. 6 Steps to generate all A- DOGE features (See Table 2)

3.5 Computational complexity

Since A- DOGE computes an embedding for each graph independently, it scales linearly with the number of graphs in the dataset, i.e., N.

We analyze the asymptotic runtime of A- DOGE on a single graph G with n nodes, m edges, and D total node attributes (including one-hot encoded labels and categorical attributes). We use the Gauss quadrature and Lanczos algorithm described by Dong *et al.* [10] to compute a (cL)DOS *histogram*. This involves (i) running η_L Lanczos iterations, each requiring $O(\eta_L(n+m))$ operations, followed by (ii) the eigendecomposition of a tridiagonal $\eta_L \times \eta_L$ matrix, with $O(\eta_L^2)$ operations. Note that although a tridiagonal matrix eigendecomposition has a quadratic worst-case complexity theoretically, this operation is extremely fast in practice—especially for real-world matrices. Each *aggregate feature* requires a dot product of two vectors of size B for $O(B)$,

where we use $2K$ different frequency response functions (i.e., $\phi(\cdot)$'s) in total (K each for Chebyshev and powers). Then, the total complexity of computing one histogram and its related aggregate features is $O(\eta_L^2 + \eta_L m + \eta_L n + KB)$. This gives a total runtime of $O\left((\eta_L^2 + \eta_L m + \eta_L n + KB) \cdot \alpha\right)$, where α denotes the number of desired graph-level features (i.e., embedding size) in A- DOGE.

Notably, A- DOGE is modular and can include any subset of the features in Table 2. For datasets with a large number of node attributes, one can skip cLDOS features, or only choose important attribute-pairs to ensure $\alpha = O(D)$. Also note that each aggregate feature for a given $\phi(\cdot)$ can be computed independently, and hence can be easily parallelized.

4 Experiments

To evaluate A- DOGE, we design both quantitative and qualitative experiments to answer the following questions.

Q1. **Graph Classification** How does A- DOGE (unsupervised) compare to the modern GNNs and graph kernels (un/supervised) on benchmark graph classification tasks?
Q2. **Exploratory Graph Analysis** Can A- DOGE provide insights for mining real-world attributed graphs?
Q3. **Efficiency** How fast and scalable is A- DOGE?
Q4. **Boosting GNNs** Can the unsupervised features generated by A-DOGE help improve the expressiveness of modern GNNs further?

Table 3 Datasets used in experiments and summary statistics

	N	Cls	Avg. n	Avg. m	Lbl	Attr
REDDIT-B	2000	2	429.6	497.7	–	–
REDDIT-5K	5000	5	508.5	594.8	–	–
COLLAB	5000	3	74.5	2457.8	–	–
IMDB-BIN	1000	2	19.8	96.5	–	–
IMDB-MUL	1500	3	13.0	65.9	–	–
DD	1178	2	284.3	715.7	78	–
PROTEINS	1113	2	39.1	72.8	–	1
AIDS	2000	2	15.7	16.2	38	4
BandPass	5000	2	200	1072.6	–	1
Congress	200	2	4196	450662.5	3	–
MIG	200	2	3075	1092282.0	19	–
Facebook100	100	n/a	12083.2	469845.4	–	7
BorderStates	49	n/a	367.4	21633.9	2	–
CountingSub	5000	4	18.8	62.6	–	–
GraphProp	7040	3	19.5	101.1	–	–

4.1 Experiment setup

Datasets. The list of all datasets used in the experiments and summary statistics are given in Table 3.

Graph classification benchmark datasets For graph classification, we use eight benchmark datasets from TUDataset repository.[3] Five are commonly used social network datasets, REDDIT-B, REDDIT-5K, COLLAB, IMDB-BIN and IMDB-MUL. These contain only plain graphs—a setting with which all the baselines are compatible.

- REDDIT-B is a balanced dataset, used in [37], where each graph is an online thread with nodes being users and edges representing the existence of direct response between two users. The task is to classify the thread (graph) as a Q&A-based community or discussion-based community. REDDIT-5K is similar to REDDIT-B but has 5 different community types.
- COLLAB [37] is a scientific collaboration dataset coming from three research fields: high energy physics, condensed matter physics, and astrophysics. Each graph represents a collaboration network with nodes being researchers and edges representing the collaborating relations.
- IMDB-BIN [37] is a movie collaboration dataset where each graph is a movie collaboration network with nodes being actors/actresses and edges representing that two actors/actresses have appeared in the same movie. The task is to classify a graph into two genres: action and romance. IMDB-MUL is a 3-class version IMDB-BIN.

The other three are biochemistry datasets, PROTEINS, DD and AIDS, which have node labels and/or attributes.

[3] https://chrsmrrs.github.io/datasets/docs/datasets/.

- PROTEINS [38] and DD [39] are two biochemistry datasets where each graph is a molecular structure of a protein. The task is to classify a graph to two categories: enzyme and non-enzyme.
- AIDS [40] contains many molecular graphs where nodes represent atoms and edges indicate the valence between two atoms. The task is to predict whether the molecular is useful for treating AIDS or not.

Graph classification band-pass datasets We also use four other graph datasets to specifically showcase the strengths of A- DOGE in leveraging the full graph spectrum.

- BandPass is a synthetic dataset consisting of images generated via sinusoidal patterns from two frequency ranges, as used in [25].
- Congress is based on the voting patterns in 41 US Senates (1927–2008), as used in [34], where nodes represent senators (labeled by party affiliation) and edge weights represent voting agreement. Nodes depicting the same senator who appear across multiple years are also connected with an edge. To create separate classes of graphs, we add noise to edge weights between same-party senators (class 1), and randomly picked senators (class 2) in one randomly picked Senate. We repeat this process to obtain 100 graphs for each class.
- Congress-L is generated by picking one Senate at random and shuffling the labels of senators via random swaps; 50 swaps in class 1, and 300 in class 2. We repeat this process to obtain 100 graphs for each class.
- MIG is based on the county-to-county migration in the USA, also used in [34]. Each node is a county (labeled using its state), and edge weights represent the amount of migration.[4] Of these, we pick two bordering states at random and add a small amount of noise to edge weights. For class 1, we add noise between same-state counties, and for class 2, we add noise between randomly picked counties. We repeat this process to obtain 100 graphs for each class.

Exploratory graph mining datasets In addition to the above datasets, we perform graph exploratory analysis using A- DOGE on two more datasets:

- Facebook100 consists of Facebook college social networks from 100 American institutions [41], with student demographic information (major, dorm, status, class-year, etc.) as node attributes.
- BorderStates is built from the MIG dataset, by inducing 49 separate graphs—one for each mainland state and its bordering states. We label counties of the selected state as 0 and the counties of the neighbors as 1.

GNN expressiveness datasets Furthermore, to additionally test whether the unsupervised features can help improve expressiveness or the separation power of graph neural networks (GNNs), we use two additional datasets with tasks related their expressiveness.

- CountingSub [42] is a simulated dataset with random graphs, and for each graph its number of triangles, tailed triangles, stars, and 4-cycles are pre-computed as target values for regression. The task is to count number of substructures for any input graph.
- GraphProp [43] is also a simulated dataset with random graphs. The task is to regress a graph to some graph-level properties, including the connectedness (binary), the diameter, and the radius of the graph.

[4] Note that the migration graph is originally directed. We create an undirected graph with edge weights w_{ij} set to $\frac{(m_{ij}+m_{ji})^2}{p_i p_j}$, where m_{ij} is the total migration from county i to county j and p_i depicts county i's total population.

These substructure counting tasks and graph property regressing tasks are closely related to expressiveness measurement of GNNs.

Baselines. We compare A- DOGE quantitatively to various unsupervised and supervised graph embedding, graph kernel, and graph neural network methods on graph classification tasks.

Unsupervised explicit graph embeddings are in the same category as A- DOGE and hence are the most comparable. As baselines from this category, we compare to

• FGSD [11], • NETLSD [12] and • G2VEC [13], which we described briefly in Sect. 1.1.

Graph kernels are also unsupervised; here, we use three of the best performing kernels on classification benchmarks, as well as a recent DOS-based graph kernel.

- WL [14]: the Weisfeiler–Lehman graph kernel,
- WL- OA [15]: the Weisfeiler–Lehman Optimal Assignment kernel,
- PK [17]: the Propagation Kernel, and
- DOSGK [19], the Density of States Graph Kernel.

GNN baselines include state-of-the-art supervised models, such as

• CHEBNET [22], • GCN [20], and • GIN [21].

Note that FGSD, NETLSD, and DOSGK are for plain graphs only. G2VEC, WL, and WL- OA admit node labels but not (continuous) attributes. Therefore, they input only the admissible parts of a graph dataset for classification.

Model configuration. In our experiments with A- DOGE, we set $\eta_L = 100$, $B = 200$ and $K = 100$ (see Table 2). For plain datasets, we use node degree as a continuous attribute. For FGSD, we use L^{-1} as the distance function and 0.001 as the binwidth. For NETLSD, we use heat trace signatures at 250 different values of t logarithmically spaced in $[10^{-2}, 10^{2}]$. For G2VEC, we set the WL iteration count to 5 and output dimension to 1024. For the kernels WL, WL- OA and PK, we use the implementation from the GraKel package,[5] and the default parameters suggested. For DOSGK, same as with A- DOGE, we use 200 bins and 100 Chebyshev moments. For all the GNNs, we use mean-pooling as the readout function. Notice that A-DOGE uses all designed features presented in Sect. 3 as input. These features can be directly input to SVM for graph classification and can be transformed to a fixed dimension embedding by a 2-layer MLP, which is then used for augmenting GNN's embedding space.

System configuration. We run all non-GNN experiments on one core of Intel(R) Xeon(R) CPU E5-2667 v3 CPU @3.20GHz. GNN experiments are run on a server with NVIDIA Tesla V100 GPU and one core of Intel(R) Xeon(R) Gold 6248 CPU @2.50GHz.

4.2 Graph classification

Classifier configurations. For classification with the embeddings produced by unsupervised methods, we use the kernel-SVM[6] classifier with the regularization parameter C chosen from $\{10^{-3}, 10^{-2}, \dots, 10^{3}\}$ via 10-fold cross-validation. We perform this experiment 10 times using random splits. For explicit embeddings, we normalize each feature, and set γ to be the inverse of the median of pairwise ℓ_2 distances between all embeddings. For A- DOGE, we also set the option of using LDOS, cLDOS features, and the option of using aggregate FRFs as hyperparameters. We normalize all kernels symmetrically. For GNNs, we train them

[5] https://ysig.github.io/GraKeL/.

[6] SVM facilitates comparable results between implicit and explicit kernels.

Table 4 Graph classification performance by A- DOGE and its DOS-only (i.e., no attributes) variant DOGE, compared with three types of baselines

	Graph embedding (unsupervised)					Graph kernels (unsupervised)				GNNs (supervised)		
	A- DOGE	DOGE	FGSD	NetLSD	G2VEC	WL	WL- OA	PK	DOSGK	ChebNet	GCN	GIN
RED-B	91.6 (1.5)	90.3 (1.8)	82.4 (2.6)	85.6 (2.2)	74.2 (2.7)	83.9 (0.5)‡	88.9 (0.1)‡	85.5 (0.3)‡	88.8 (0.3)*	90.2 (2.0)	89.9 (2.0)	**91.7** (1.6)
RED-5K	**55.6** (2.2)	53.8 (2.1)	47.0 (1.8)	45.9 (2.1)	41.5 (1.6)	51.2 (0.3)*	E	E	52.8 (0.2)*	55.0 (2.2)	54.2 (1.7)	54.7 (2.0)
COLLAB	72.2 (2.0)	72.2 (2.0)	70.2 (1.8)	68.4 (1.9)	57.9 (1.5)	74.8 (0.2)*	79.8 (1.6)	77.8 (1.7)	80.8 (0.2)*	**84.6** (1.1)	84.2 (1.2)	83.8 (1.6)
IMDB-B	72.6 (4.3)	71.6 (4.3)	70.6 (4.1)	69.7 (4.1)	56.0 (4.1)	71.3 (1.0)‡	73.5 (0.6)	71.2 (0.7)‡	72.8 (0.9)*	80.2 (3.9)	79.9 (3.7)	**80.8** (4.5)
IMDB-M	47.8 (3.5)	47.6 (3.7)	48.6 (3.4)	47.9 (3.7)	44.4 (3.8)	50.7 (0.6)‡	50.7 (0.5)‡	51.0 (0.7)‡	49.4 (0.5)*	55.6 (2.7)	55.2 (2.7)	**56.3** (3.1)
DD	80.1 (3.5)	76.2 (3.4)	76.5 (3.5)	76.6 (3.5)	76.2 (3.5)	80.9 (0.3)	79.9 (0.5)	**81.6** (0.5)	73.4 (3.7)	78.9 (1.9)	78.0 (1.8)	79.3 (1.9)
PROTN	74.9 (3.5)	74.9 (3.5)	74.2 (3.3)	74.5 (4.0)	72.1 (3.1)	73.9 (0.7)‡	75.9 (0.6)‡	74.6 (0.5)‡	72.1 (3.9)	78.3 (2.7)	76.7 (3.5)	**78.4** (3.9)
AIDS	**99.8** (0.3)	99.8 (0.3)	99.6 (0.4)	99.6 (0.5)	98.8 (0.7)	99.7 (0.0)‡	99.7 (0.0)‡	99.7 (0.0)‡	99.1 (0.7)	96.9 (1.6)	95.5 (1.3)	98.6 (0.6)
Cong	**99.5** (1.5)	54.7 (11.0)	95.1 (4.3)	99.5 (1.5)	86.8 (7.4)	84.8 (7.3)	81.1 (7.7)	68.6 (8.3)	60.0 (10)	50.0 (0.0)	50.0 (0.0)	57.0 (5.9)
Cong-1	**78.0** (8.6)	58.9 (10.0)	50.0 (0.0)	60.4 (9.7)	59.8 (11)	62.2 (10)	62.3 (10)	58.2 (10)	55.7 (9.7)	50.0 (0.0)	50.0 (0.0)	71.5 (9.4)
MIG	**100.0** (0)	62.3 (9.7)	99.5 (1.5)	99.9 (1.1)	50.0 (0.0)	99.8 (1.4)	99.8 (1.4)	100 (0.0)	53.5 (12)	100.0 (0.0)	78.5 (1.7)	100.0 (0.0)
BPass	90.8	51.9	47.9	51.4	50	50	51.6	70.4	48.5	**98.2**†	77.9†	87.6†
Avg	82.5	69.1	74.1	75.8	66.0	75.6	76.6	76.3	68.6	78.4	74.2	80.5

The highest performance per dataset is **in bold**, and the highest among unsupervised methods is underlined. E denotes the code outputting an error; The numbers with symbols denote the paper from which the numbers are taken: ‡[24], * [19], †[25]

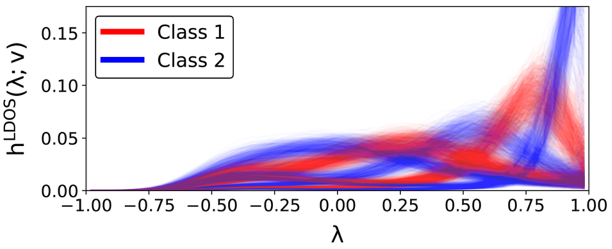


Fig. 7 LDOS histograms for all graphs in BandPass, plotted as lines. Each class can be characterized by specific bands of eigenvalues

end-to-end using cross-entropy loss, and hyperparameters (learning-rate at 0.005, layers in {2,3,5,7}, hidden sizes from {32,64,128} and epochs up to 200) selected via 10-fold cross-validation. For each of the above methods, we report the mean test accuracy for the best choice of hyperparameters, and the corresponding standard deviation on every dataset except BandPass, for which we use the single train-validation-test split as specified in [25].

Results. Table 4 contains all the performance results of our classification experiments. Among the benchmark datasets, A- DOGE achieves on par performance with the most competitive unsupervised baselines and is often comparable to (supervised) GNNs, while being considerably more resource-frugal.

On the other four datasets, A- DOGE significantly outperforms all baseline methods due to its ability to capture the alignment of labels and attributes with graph structure at a multi-scale level, even in databases with as few as 200 graphs. Provided A- DOGE uses considerably lower resources in comparison with kernels and GNNs, and considering that the latter are trained end-to-end, we do not expect A- DOGE to exhibit state-of-the-art performance on every dataset. Still, A- DOGE outperforms/equals all baselines on 7 of the datasets. Moreover, A- DOGE stands out as the top choice based on average performance across all datasets.

On the BandPass dataset, only the spectrally designed CHEBNET is able to outperform A- DOGE. This can be attributed to the way that BandPass is created, wherein graph classes are formed based on the frequency band used to generate the underlying image. Figure 7 depicts the LDOS histograms of the graphs in the BandPass dataset. We can clearly see that capturing specific bands of the eigenspectrum suffices to characterize the disparity between the two graph classes.

Feature ablation. Table 4 also shows the DOS-only version of A- DOGE without using node labels and attributes, called DOGE. We observe that in the benchmark datasets, graph structure seems to hold most of the useful information needed for classification, and hence, there is only a small improvement in performance from using node attributes. In the rest of the datasets, node attributes play an important role, causing significant improvements in results for A- DOGE by using LDOS and cLDOS features.

4.3 Graph data mining

To demonstrate the interpretability of the A- DOGE features, we perform exploratory graph analysis on three real-world datasets, Facebook100, Congress and BorderStates.

Facebook100. In Facebook100, we denote each categorical feature (e.g., major) with its one-hot encoding, and hence, each particular value (e.g., Computer Science) has its own (binary) attribute vector. We first visualize the Facebook100 graphs via LDOS

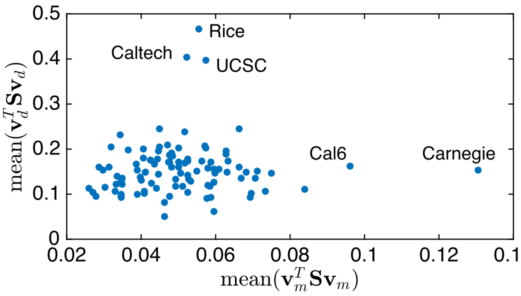
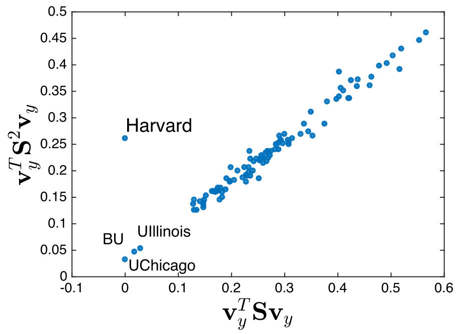


Fig. 8 Average homophily w.r.t. *major* vs. *dorm* in 100 Facebook college social networks in the USA, where $\mathbf{v}_m$ and $\mathbf{v}_d$, respectively, refer to an attribute vector corresponding to a particular major m and dorm d



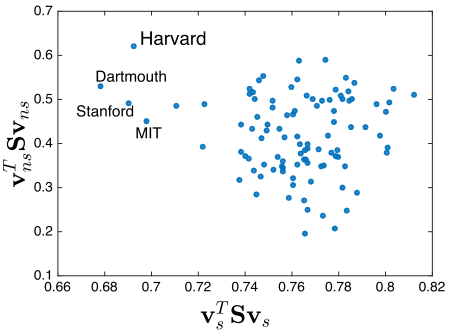


Fig. 9 (left) Homophily w.r.t. *class_year* based on $k = 1$ and $k = 2$-length paths over all 100 colleges. $\mathbf{v}_y$ refers to continuous attribute vector with class years. (right) Homophily within student and non-student communities in all 100 colleges. Binary vector $\mathbf{v}_s$ ($\mathbf{v}_{ns}$) depicts student (non-student) *status*

aggregate features using these attribute vectors, with small positive power functions as FRF to capture the assortativity (homophily) of different attributes across different college networks. In each graph, we compute the aggregate feature that estimates $\mathbf{v}_m^T\mathbf{S}\mathbf{v}_m$ for every major captured by $\mathbf{v}_m$, and similarly $\mathbf{v}_d^T\mathbf{S}\mathbf{v}_d$ for every dormitory captured by $\mathbf{v}_d$. Figure 8 plots the mean homophily with respect to *major* and *dorm* for each of the 100 colleges.

While Carnegie pops up as having the highest correlation between edges and students with the same *major*, comparing the ranges of both axes suggests that *dorm* is a much stronger indicator of students within a college being friends. Moreover, this tendency seems to be more pronounced in Rice, Caltech and UCSC. This is also backed up by findings in [41] and the real-world knowledge that Rice and Caltech are organized predominantly by dorms and other on-campus housing.

We also analyze similar aggregate functions over the continuous attributes. Figure 9(left) plots the assortativity with respect to *class_year* for $k = 1$ and $k = 2$ for the power functions, which capture 1- and 2-length paths. As we expect, these features are highly correlated in most colleges—with the striking exception of Harvard, where it appears that 2-length paths are common between individuals of similar *class_year*, but this is not the case with 1-length paths. To investigate further, we plot homophilies for student and non-student populations for all colleges in Fig. 9(right) and we learn that the Harvard network consists of a comparatively higher number of edges amongst non-student members, most of whom have empty or very disparate *class_year*. Even if edges between students are fewer, this is corrected when we look at 2-length paths instead.

`Congress`. Next, we want to explore scenarios where *interactions between attributes* prove important to understanding properties of a graph. To this end, we look at the `Congress`

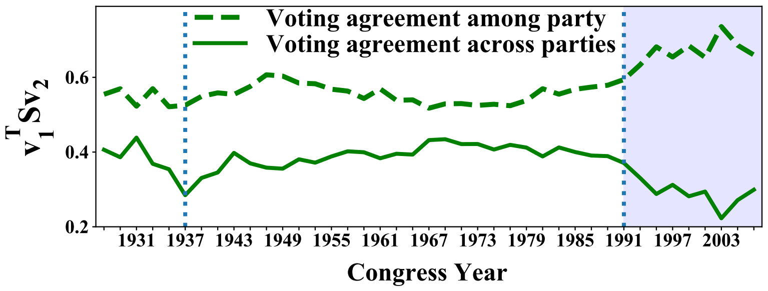


Fig. 10 Voting agreement within (dashed curve, $\mathbf{v}_1 = \mathbf{v}_2 = \mathbf{v}_d$ or $\mathbf{v}_r$) and across (solid curve, $\mathbf{v}_1 = \mathbf{v}_d$, $\mathbf{v}_2 = \mathbf{v}_r$) political parties over the years, for 41 Senates during 1927–2008

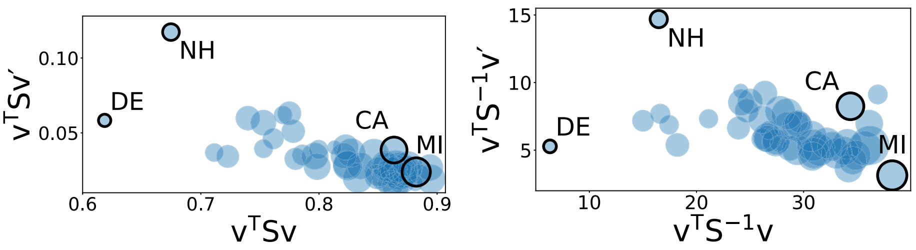


Fig. 11 Comparison of migration patterns for each US state—within its counties vs. across its borders; migration (left) over a local range, and (right) on a global scale. $\mathbf{v}_w$ and $\mathbf{v}_b$ refer to binary vectors denoting *w*ithin and *b*order-state counties, respectively. Node sizes correlate to size of state

graph, where the two attribute vectors are binary vectors $\mathbf{v}_d$ and $\mathbf{v}_r$ corresponding to Democrat and Republican senators, respectively (ignoring the small minority of independents). We plot within-party agreement $(\mathbf{v}_d^T \mathbf{S} \mathbf{v}_d + \mathbf{v}_r^T \mathbf{S} \mathbf{v}_r)/2$ and cross-party agreement $(\mathbf{v}_d^T \mathbf{S} \mathbf{v}_r)$ over the years in Fig. 10.

We can observe that beginning from the 1990s, senators tend to agree among their parties, and disagree with the opposite party to a higher extent, hinting at a growing polarization in politics. We note that agreement across parties is also low in 1937 (see the "dip"); however, this is better explained by the fact that this congress had overwhelmingly more number of democrats. There is no hint of polarization for that instance, since there is no corresponding rise in the dashed (within-party) curve. Figure 10 shows that aggregate functions from A-DOGE not only help us observe such phenomenon but also help quantify them to a relative extent.

`BorderStates`. Lastly, we analyze `BorderStates`, comparing within-state migration against cross-border migration for each of the 49 mainland states in the USA. We focus on LDOS aggregate features; this time using both positive and negative power functions in order to analyze both short and long-range migration patterns. In other words, while small positive powers (e.g., $k = 1$) capture local migration patterns, negative powers (e.g., $k = -1$) capture paths of all lengths and thereby reflect long-range migration behavior on a relatively global scale.

From Fig. 11(left), we observe that at the local scale, most states have greater within-state migration than cross-border migration. Comparatively, NH and DE, being the states with the least number of counties (10 and 3, respectively), exhibit lower within-state migration. Moreover, due to NH's geographical and political similarity with its bordering states, it shows highest cross-border migration. On the other hand, larger states such as CA and

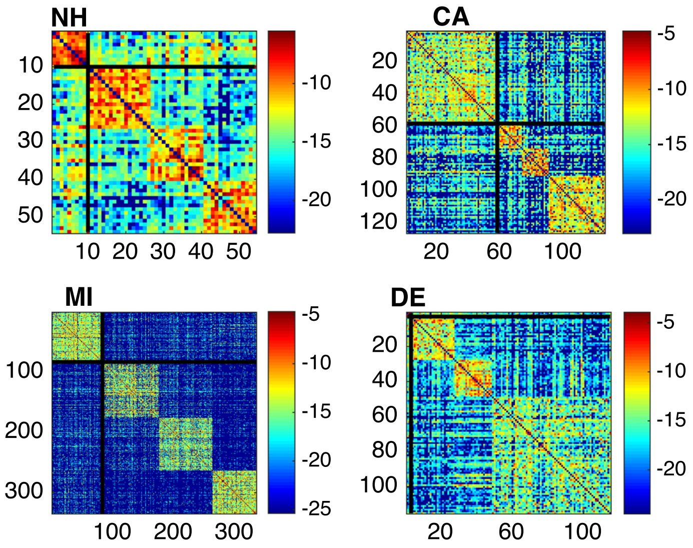


Fig. 12 Migration graph of 4 states—NH (New Hampshire), CA (California), MI (Michigan), and DE (Delaware)—from `BorderStates`, depicting migration volume (in logarithm) within state as well as across-border states. Black lines separate counties (10, 58, 89, and 3, respectively, for NH, CA, MI, and DE) of the state from those in border states

MI exhibit mostly within-state migrations on the local scale. However, on the global scale (Fig. 11(right)), the difference between these is more pronounced, since CA is a more popular long-range migration destination than MI. The ratio between the average within-state and average across-border migration is 78.68 for MI—much larger as compared to DE, CA, and NH with values 43.43, 34.40, and 14.17, respectively.

Figure 12 helps explain this observed behavior visually, where we show via heatmaps the total migration volume among the counties of each state as well as the counties in their immediate neighbor/border states. DE and NH are small states with only a few counties, which explains the small within-state migration volume in Fig. 11. On the other hand, NH exhibits the highest cross-border traffic as compared to the others. In stark contrast to NH, MI has relatively much less border traffic as compared to within-state migration. CA, on the other hand, exhibits large-volume migration both within-state and across-state.

4.4 Efficiency

A- DOGE is not directly comparable to all the baselines in terms of resource requirements. GNNs and G2VEC need GPU processing, which make them incomparable to CPU-based A-DOGE and the rest. Other differences, such as supervised training and collective processing of the graphs via multiple passes over the dataset (in contrast to one-by-one/independent processing by A- DOGE) put them in a different "league".

On the other hand, kernel baselines need considerably more memory. WL, WL- OA and PK compute intermediate data (e.g., compressed labels) based on all the graphs in memory. These and DOSGK produce a $N \times N$ kernel matrix that is also memory-resident.

FGSD and NETLSD are comparable in the sense that, similar to A- DOGE, they process the graphs independently one-by-one. Likewise, they are also unsupervised. However, they

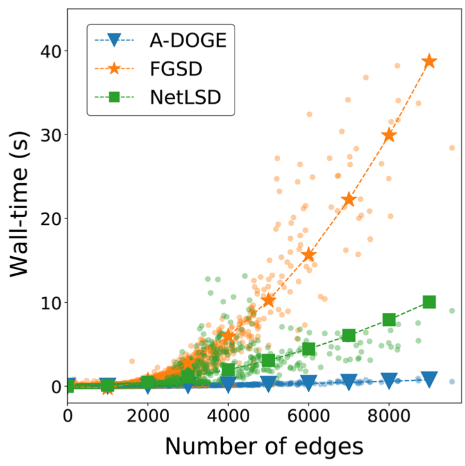


Fig. 13 Runtimes per graph in the `REDDIT-5K` dataset for each of the A- DOGE and the two baselines which can compute embeddings independently

cannot handle node labels/attributes. Nevertheless, we provide running time and scalability comparison in Fig. 13 that plots the runtime vs. size in number of nodes for individual graphs in the `REDDIT-5K` dataset. For any graph from the dataset (up to 9500 edges), A-DOGE does not take more than 1 s to compute. Figure 1 compares this runtime for 3 of our largest datasets. We can see that A- DOGE achieves the best time-accuracy trade-off among competing baselines. For methods with comparable or better accuracy scores (e.g., GIN), A- DOGE is almost twice as fast on average. For baselines with similar runtime (e.g., WL), A- DOGE achieves significantly higher accuracy.

4.5 Boosting GNNs

Modern GNNs have achieved significant success in both node-level and graph-level tasks, with applications widely existing in many domains such as drug discovery, social network analysis, image analysis and bioinformatics [44–46]. However, the widely used message-passing-based graph neural networks are known to have limited expressiveness and are specifically upper bounded by the first-order Weisfeiler–Leman [21]. Many research works have been designed to improve the expressiveness of GNNs, and one direction is to use additional structural features [47, 48] or even random features [49, 50] to augment the input of GNNs, which achieves noticeable improvement over many tasks.

In this section, we investigate whether the unsupervised features from A-DOGE can help improve the expressiveness of GNNs. We focus on two categories of tasks related to expressiveness: (1) counting the number of substructures and (2) regressing various graph-level properties. To this end, we use two datasets, `CountingSub` and `GraphProp`, as introduced in Sect. 4.

Model configurations. As the two datasets contain only plain graphs without node and edge attributes,[7] we only compute DOS and PDOS histograms from A-DOGE and augment them into any GNN model. The architecture of GNN contains the stacking of multiple message-passing layers that update node features, and a following pooling layer that aggregates all nodes into a graph-level feature. We augment DOS and PDOS into GNN as follows:

[7] Node degree is used as a structural node feature as input.

Table 5 Comparison between GNN with and without using DOS (graph-level) and PDOS (node-level) features. (MAE: mean absolute error)

Method	Counting substructures (MAE)				Graph properties ($\log_{10}$(MAE))		
	Triangle	Tailed Tri	Star	4-Cycle	IsConnected	Diameter	Radius
GCN	0.4186	**0.3248**	**0.1798**	0.2822	−1.7057	−2.4705	−3.9316
+ DOS/PDOS	**0.3198**	0.3360	0.3623	**0.2784**	**−1.8511**	**−3.3207**	**−4.1816**
GIN	0.3569	0.2373	0.0224	0.2185	−1.9239	**−3.3079**	**−4.7584**
+ DOS/PDOS	**0.1355**	**0.1430**	**0.0260**	**0.0981**	**−1.9691**	−3.2374	−4.6507

- DOS involves graph-level features. We pass DOS into a MLP and add the transformed DOS to the final layer of a GNN (right after pooling layer).
- PDOS involves node-level features. We concatenate the original node features (i.e., degree) of the graph with the PDOS before input to a GNN.

In the experiments we focus on two types of GNN: GCN [20], and GIN [21]. For hyperparameters, we use 6 layers and hidden size 128. Batch normalization is applied after each layer. We use Adam as optimizer with learning rate 0.001. For both DOS and PDOS the histogram's number of bins is set as 100.

The results are shown in Table 5. For both GIN and GCN, adding DOS and PDOS helps boost the performance of counting substructures as well as regressing graph properties dramatically in most cases. The results support that the DOS and PDOS contain additional information that cannot be extracted by GNNs; hence, adding them enhances the expressive power of GNNs. Graph eigenspectrum captures important structural graph properties like the diameter, connectedness, clustering, etc. [9]. DOS and LDOS contain rich spectral information that are thus helpful for characterizing a graph.

Characterizing the expressiveness of DOS and LDOS theoretically is an interesting direction which we aim to investigate in the future. In fact, Huang et al. [19] proved that LDOS contains all information regarding the return probabilities for each node. A broader research question is to discover the relationship between the graph spectrum and the graph isomorphism test, which historically has only been studied under certain conditions [51].

5 Conclusion

We propose A- DOGE, an unsupervised graph embedding technique designed to efficiently capture structural properties as well as node labels and attributes of a graph. To this end A- DOGE uses spectral density, or density of states (DOS), derived from the eigenspectrum of the graph, as a tool to capture both global and local properties of a graph. Further, we extend local density of states to leverage node labels and attributes, and capitalize on fast approximation algorithms making A- DOGE efficient and scalable to large graphs both in terms of time and space. Being unsupervised, it is not only suitable for downstream supervised graph classification tasks, but also applies well to exploratory graph analysis. Through both quantitative and qualitative experiments, we show the efficacy and efficiency of A- DOGE, where it outperforms unsupervised baselines and performs comparably to the supervised GNNs on graph classification tasks, and provides various insights into the analysis of real-world attributed graphs.

Acknowledgements This work is sponsored by NSF CAREER 1452425. We also thank PwC Risk and Regulatory Services Innovation Center at CMU. Any conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding parties.

Funding Open Access funding provided by Carnegie Mellon University.

References

1. Wale N, Watson IA, Karypis G (2008) Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl Inf Syst 14(3):347–375
2. Przulj N (2010) Biological network comparison using graphlet degree distribution. Bioinform 26(6):853–854
3. Duen H, Carey N, Jeffrey W, Adam W, Christos F (2011) Polonium: tera-scale graph mining for malware detection. In: SIAM SDM
4. Ribeiro B, Chen N, Kovacec A (2019) Shaping graph pattern mining for financial risk. Neurocomputing 326:123–131
5. Mieghem PV (2011) Graph spectra for complex networks. Cambridge University Press, Cambridge
6. Pothen A, Simon HD, Liou K-P (1990) Partitioning sparse matrices with eigenvectors of graphs. SIAM J Matrix Anal Appl 11(3):430–452
7. Fill JA (1991) Eigenvalue bounds on convergence to stationarity for nonreversible markov chains, with an application to the exclusion process. Ann Appl Probab 62–87
8. Chakrabarti D, Wang Y, Wang C, Leskovec J, Faloutsos C (2008) Epidemic thresholds in real networks. ACM TISSEC 10(4):1–26
9. Jin S, Zafarani R (2020) The spectral zoo of networks: embedding and visualizing networks with spectral moments. In: KDD, pp 1426–1434
10. Dong K, Benson AR, Bindel D (2019) Network density of states. In: KDD, pp 1152–1161
11. Verma S, Zhang Z-L (2017) Hunt for the unique, stable, sparse and fast feature learning on graphs. In: NIPS, pp 88–98
12. Tsitsulin A, Mottin D, Karras P, Bronstein A, Müller E (2018) Netlsd: hearing the shape of a graph. In: KDD, pp 2347–2356
13. Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S (2017) graph2vec: learning distributed representations of graphs. arXiv:1707.05005
14. Shervashidze N, Schweitzer P, van Leeuwen EJ, Mehlhorn K, Borgwardt KM (2011) Weisfeiler–Lehman graph kernels. J Mach Learn Res 12:2539–2561
15. Kriege NM, Giscard P-L, Wilson RC (2016) On valid optimal assignment kernels and applications to graph classification. In: NIPS, pp 1615–1623
16. Wu L, Zhang Z, Nehorai A, Zhao L, Xu F, Learning AS (2019) Sage: Scalable attributed graph embeddings for graph classification. In: ICLR workshop on representation learning on graphs and manifolds
17. Neumann M, Garnett R, Bauckhage C, Kersting K (2016) Propagation kernels: efficient graph kernels from propagated information. Mach Learn 102(2):209–245
18. Zhang Z, Wang M, Xiang Y, Huang Y, Nehorai A (2018) RetGK: graph kernels based on return probabilities of random walks. In: NeurIPS, pp 3968–3978
19. Huang L, Graven AJ, Bindel D (2021) Density of states graph kernels. In: SDM, pp 289–297. SIAM
20. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR
21. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: ICLR, pp 1–17
22. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS, pp 3837–3845
23. Levie R, Monti F, Bresson X, Bronstein MM (2019) CayleyNets: graph convolutional neural networks with complex rational spectral filters. IEEE Trans Sign Process 67(1):97–109

24. Kriege NM, Johansson FD, Morris C (2020) A survey on graph kernels. Appl Netw Sci 5(1):6
25. Balcilar M, Guillaume R, Héroux P, Gaüzère B, Adam S, Honeine P (2021) Analyzing the expressive power of graph neural networks in a spectral perspective. In: ICLR
26. Xu K, Hu W, Leskovec J, Jegelka S (2019) How powerful are graph neural networks? In: ICLR
27. Wang F, Landau DP (2001) Efficient, multiple-range random walk algorithm to calculate the density of states. Phys Rev Lett 86(10):2050
28. Li C, Sra S, Jegelka S (2016) Gaussian quadrature for matrix inverse forms with applications. In: International conference on machine learning. PMLR, pp 1766–1775
29. Golub GH, Welsch JH (1969) Calculation of gauss quadrature rules. Math Comput 23(106):221–230
30. Golub GH, Meurant G (1997) Matrices, moments and quadrature ii; how to compute the norm of the error in iterative methods. BIT Numer Math 37(3):687–705
31. Farkas IJ, Derényi I, Barabási A-L, Vicsek T (2001) Spectra of real-world graphs: beyond the semicircle law. Phys Rev E 64(2):026704
32. Banerjee A, Jost J (2008) Spectral plot properties: towards a qualitative classification of networks. Netw Heterog Media 3(2):395
33. McGraw PN, Menzinger M (2008) Laplacian spectra as a diagnostic tool for network structure and dynamics. Phys Rev E 77(3):031102
34. Cucuringu M, Mahoney MW (2011) Localization on low-order eigenvectors of data matrices. arXiv:1109.1355
35. Mitrović M, Tadić B (2009) Spectral and dynamical properties in classes of sparse networks with mesoscopic inhomogeneities. Phys Rev E 80(2):026123
36. Meurant G (2007) Matrices, moments, and quadrature. In: Milestones in matrix computation: the selected works of Gene H. Golub with commentaries, p 380
37. Yanardag P, Vishwanathan S (2015) Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1365–1374
38. Borgwardt KM, Ong CS, Schönauer S, Vishwanathan S, Smola AJ, Kriegel H-P (2005) Protein function prediction via graph kernels. Bioinformatics 21(suppl-1):47–56
39. Dobson PD, Doig AJ (2003) Distinguishing enzyme structures from non-enzymes without alignments. J Mol Biol 330(4):771–783
40. Riesen K, Bunke H (2008) Iam graph database repository for graph based pattern recognition and machine learning. In: Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR), pp 287–297. Springer
41. Traud AL, Mucha PJ, Porter MA (2012) Social structure of facebook networks. Physica A 391(16):4165–4180
42. Zhengdao C, Lei C, Soledad V, Bruna J (2020) Can graph neural networks count substructures? Adv Neural Inf Process Syst 33:10383–10395
43. Corso G, Cavalleri L, Beaini D, Liò P, Veličković P (2020) Principal neighbourhood aggregation for graph nets. Adv Neural Inf Process Syst 33:13260–13271
44. Duvenaud DK, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional networks on graphs for learning molecular fingerprints. Adv Neural inf Process Syst 28
45. Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. In: The world wide web conference, pp 417–426
46. Shi L, Zhang Y, Cheng J, Lu H (2019) Skeleton-based action recognition with directed graph neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7912–7921
47. Bouritsas G, Frasca F, Zafeiriou S, Bronstein MM (2020) Improving graph neural network expressivity via subgraph isomorphism counting. arXiv:2006.09252
48. Barceló P, Geerts F, Reutter JL, Ryschkov M (2021) Graph neural networks with local graph parameters. Adv Neural Inf Process Syst 34:25280–25293
49. Sato R, Yamada M, Kashima H (2021) Random features strengthen graph neural networks. In: Proceedings of the 2021 SIAM international conference on data mining (SDM), pp 333–341. SIAM
50. Abboud R, Ceylan İİ, Grohe M, Lukasiewicz T (2021) The surprising power of graph neural networks with random node initialization. In: Proceedings of the thirtieth international joint conference on artifical intelligence (IJCAI)
51. Babai L, Grigoryev DY, Mount DM (1982) Isomorphism of graphs with bounded eigenvalue multiplicity. In: Proceedings of the fourteenth annual ACM symposium on theory of computing, pp 310–324

Lingxiao Zhao received the B.E. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2016, and the M.S. degree in 2018 in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, where he is currently working toward the Ph.D. degree in Machine Learning joint Public Policy with Heinz College and Machine Learning Department. His research interests include deep learning on graphs and many applications with graph structured data.

Saurabh Sawlani is a research engineer at SoundHound Berlin, specializing in Machine Learning algorithms for Voice AI. He holds a PhD in Algorithms Combinatorics and Optimization from Georgia Institute of Technology.

Leman Akoglu is the Heinz College Dean's Associate Professor of Information Systems at Carnegie Mellon University. She has also received her Ph.D. from CSD/SCS of Carnegie Mellon University in 2012. Dr. Akoglu's research interests broadly span machine learning and data mining, and specifically graph mining, pattern discovery and anomaly detection, with applications to fraud and event detection in diverse real-world domains. Dr. Akoglu is a recipient of the SDM/IBM Early Career Data Mining Research award (2020), National Science Foundation CAREER award (2015) and US Army Research Office Young Investigator award (2013). Her early work on graph anomalies has been recognized with the Most Influential Paper (PAKDD 2020), which was awarded the Best Paper (PAKDD 2010). Her research has been supported by the NSF, US ARO, DARPA, Adobe, Capital One Bank, Facebook, Northrop Grumman, PNC Bank, PwC, and Snap Inc.