DINE: Dimensional Interpretability of Node Embeddings

Simone Piaggesi, Megha Khosla, André Panisson, and Avishek Anand

***Abstract*—Graph representation learning methods, such as node embeddings, are powerful approaches to map nodes into a latent vector space, allowing their use for various graph learning tasks. Despite their success, these techniques are inherently black-boxes and few studies have focused on investigating local explanations of node embeddings for specific instances. Moreover, explaining the overall behavior of unsupervised embedding models remains an unexplored problem, limiting global interpretability and debugging potentials. We address this gap by developing human-understandable explanations for latent space dimensions in node embeddings. Towards that, we first develop new metrics that measure the global interpretability of embeddings based on the marginal contribution of the latent dimensions to predicting graph structure. We say an embedding dimension is more interpretable if it can faithfully map to an understandable sub-structure in the input graph - like community structure. Having observed that standard node embeddings have low interpretability, we then introduce DINE (Dimension-based Interpretable Node Embedding). This novel approach can retrofit existing node embeddings by making them more interpretable without sacrificing their task performance. We conduct extensive experiments on synthetic and real-world graphs and show that we can simultaneously learn highly interpretable node embeddings with effective performance in link prediction and node classification.**

***Index Terms*—Interpretability, node embeddings, representation learning, link prediction.**

I. Introduction

NODE embeddings are general purpose low-dimensional, continuous vertex representations in dense vector spaces. These embeddings are typically learned by trying to optimize a user-defined or flexible notion of structural similarity between vertices. Node embeddings have proven to be mature and popular techniques with widespread applications in web and social network analysis tasks like link prediction and community detection to name a few [1] due to their simplicity, and expressive power. However, one of their shortcomings is the innate lack of interpretability of the latent vector spaces they exist in. Specifically, each of the learned latent dimensions does not have a corresponding realizable interpretation in the input graphs [2], [3], [4]. This paper aims to fill this critical gap by proposing a method to retrofit "already learned" non-interpretable embeddings into a new and interpretable vector space without compromising the task performance.

The meaning of individual embedding dimensions is hard to define and determine [2], [3], [4]. We operate on a general, yet powerful notion of grounding the interpretation of each dimension to understandable sub-structures of the input graphs, e.g., communities, subgraphs, etc. This design decision has clear advantages in downstream tasks where sub-graphs or communities are clear explanations of a predictive task. As a concrete example, in a link prediction task, the likelihood of a link is higher for a pair of nodes in the same community [5]. Similarly, in several bio-medical tasks that use embedding features like [6], [7], subgraphs refer to a protein or genetic pathways. Therefore, automatically grounding latent dimensions to sub-graphs and community structures will improve understanding of the prediction process.

Existing literature investigating the interpretability of node embeddings is limited to three major aspects. First, feature-attribution methods like [8], [9], [10] return *local* explanations for single node decisions in terms of (a) subset of node features or (b) nodes/edges in the query node's local neighborhood. If embeddings are used as features, subsets of latent features are still non-interpretable. Explaining a prediction in terms of node's neighborhood is a useful first step, but these approaches cannot be used globally. Specifically, *global* explanations should describe the meaning of latent dimensions, that cannot be captured by local methods. Second, in the presence of external node labels [4], [11] measure the interpretability of embedding dimensions in terms of their global association with these labels without providing an explicit explanation like our approach. These approaches pre-suppose a certain interpretable mapping with existing categories and are not flexible. Finally, [12] searches for interpretable subspaces related to concepts from knowledge bases. Our work distinguishes itself from previous research in the following crucial ways. First, we operate on a more generalized setting extracting explanations for node embeddings that are agnostic to ground-truth labels. Second, we are the first work to explain directly the latent dimensions of the node embeddings without using any proxy tasks or labels. We are different from local explainability approaches which only explains the learnt

Manuscript received 24 August 2023; revised 6 June 2024; accepted 22 June 2024. Date of publication 9 July 2024; date of current version 13 November 2024. The work of Simone Piaggesi was supported in part by the European Community Program under Grant n.871042, in part by "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics" under Grant 834756, and in part by "XAI: Science and technology for the eXplanation of AI decision making." Recommended for acceptance by X. Chen. *(Corresponding author: Simone Piaggesi.)*

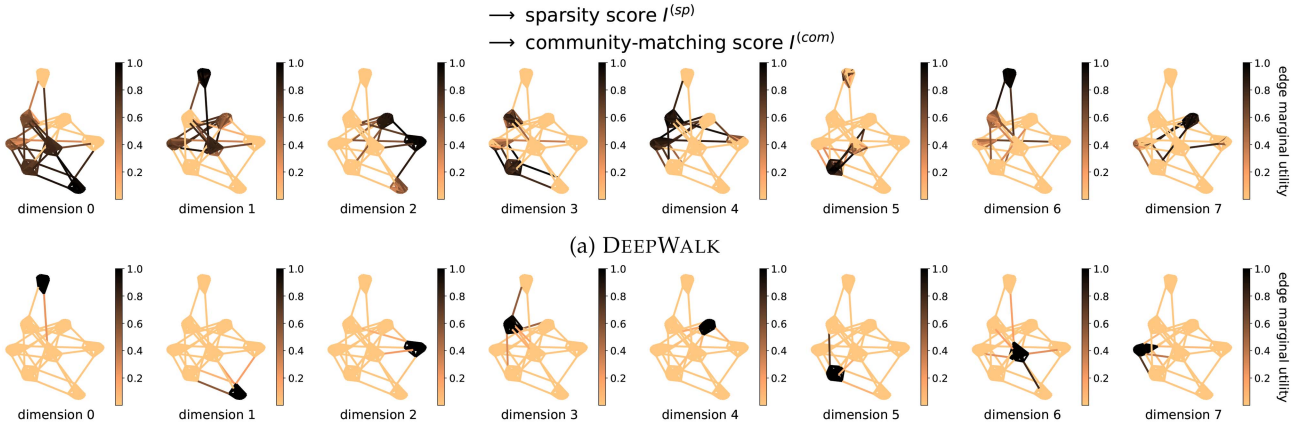
Simone Piaggesi is with the University of Pisa, 56126 Pisa, Italy (e-mail: simone.piaggesi@di.unipi.it).

Megha Khosla and Avishek Anand are with the TU Delft, 2628 CD Delft, The Netherlands.

André Panisson is with the CENTAI Institute, 10138 Torino, Italy.

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2024.3425460, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2024.3425460



(a) DEEPWALK

(b) DINE

Fig. 1. Saliency plots displaying edge marginal utilities (normalized between 0 and 1) for 8-dim node embeddings of graph generated via Stochastic Block Model [16], with 80 nodes divided into 8 cliques. For each panel, dimensions are arranged from left to right, ordered by increasing alignment with individual cliques and by increasing sparsity of the per-dimension subgraphs. In picture (a), DEEPWALK dimensions struggle to align with distinct cliques and fail to get consistently sparse explanatory structures, rendering the embeddings less interpretable. In picture (b), our approach DINE achieves interpretable dimensions that effectively match communities and exhibit high levels of sparsity.

embeddings for a particular node. Last and more importantly, we propose methods that retrofit existing node embeddings to make their dimensions more interpretable.

In this work, the central aim is finding the human-interpretable meaning of node embedding dimensions and associating latent directions with understandable structural features of the input graph. Despite this post-hoc approach, we cannot rely on existing tools for interpreting prediction models [13], [14], mainly because they are designed for supervised tasks. Instead, we analyze unsupervised node embeddings to find evidence for interpretable latent units associated with individual semantic concepts of the input data. Since many empirical graphs are characterized by an underlying community structure [15], we associate communities as interpretable semantic concepts of the input data. We first develop a metric for interpreting each dimension of the node embeddings in terms of its utility in predicting edges in the graph.

Our utility measure -$\mu_d(\mathbf{u}, \mathbf{v})$- is based on feature removal methods and expresses the individual contribution of dimension d for predicting edge (u, v) from embeddings. Using this measure, we can construct saliency maps (refer to Fig. 1), to recognize groups of edges (salient subgraphs) that are reconstructed by specific dimensions. We then define quantitative metrics to estimate the interpretability of latent dimensions: we assess interpretability as a "degree of association" with individual graph communities and the sparsity level for these associations. As an example, in Fig. 1(a), we sort DEEPWALK dimensions according to our metrics of interpretability, showing that the majority of units are not immediately interpretable since they do not match with a single clique of the synthetic graph.

Second, we propose a novel and modular approach, called DINE, to post-process existing node representations and enhance their interpretability. DINE embeds input embeddings into a new sparse, interpretable, and low-entropy vector space. Fig. 1(b) shows the result of this post-processing step on DEEPWALK vectors. DINE also preserves the topological graph information to be employed in usual downstream tasks with minimal performance loss.

In our extensive experimental evaluations, we compare our approach with other embedding methods in terms of interpretability, downstream tasks performance, and scalability over multiple real-world graph datasets. Our results show that DINE, under most experimental conditions, convincingly outperforms existing baselines in terms of dimensional interpretability with negligible to no performance losses. To summarize, our main contributions are as follows:

- We formalize the desirable properties for global explanations of node embeddings, namely *Decomposability, Comprehensibility, and Sparsity*.
- We introduce a new utility measure that allows the extraction of explanatory subgraphs, one for each dimension (addressing the property of Decomposability). Our measure is grounded in feature attribution techniques like the Shapley value, which are widely used for importance-based explanations. Moreover, we propose two metrics to measure Comprehensibility and Sparsity of explanatory subgraphs.
- We propose a novel, modular, and theoretically sound method DINE that intends to retrofit existing node embeddings to improve their global interpretability.
- We run extensive experimental analyses to establish clear gains in the interpretability-performance trade-offs using DINE.

Our code and artefacts are made available at https://www.github.com/simonepiaggesi/dine.

II. PRELIMINARIES AND RELATED WORK

A. Preliminaries and Notation

Given an undirected, unattributed and unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node embeddings are the output of an encoder function

$$\text{enc} : v \in \mathcal{V} \mapsto \text{enc}(v) = \mathbf{v} \in \mathbb{R}^D \tag{1}$$

which map nodes into geometric points of the $D-$dimensional vector space $\mathbb{R}^D$ (usually $D << |\mathcal{V}|$). We will refer to both $\text{enc}(v)$ and $\mathbf{v}$ as the embedding vectors of node $v \in \mathcal{V}$ mapped

through the encoder enc, and with v_d as entries of these vectors corresponding to dimension d. Node embeddings are collected as column vectors of the embedding matrix $\mathbf{X}(\mathcal{G}) \in \mathbb{R}^{D\times|\mathcal{V}|}$, i.e., $\mathbf{X}_{:,v} = \mathbf{v}$. Later we will refer to D as the cardinality of the set $\mathcal{D} = \{1 \dots D\}$ containing the enumerated dimensions. In Table A1 of the Appendix, available online, we provide a summary of the notation used for this paper.

In the case of DEEPWALK [17] and NODE2VEC [18], the encoder is a lookup function where node representations are learned through the optimization of a neighborhood reconstruction loss. Specifically, the output of a decoder dec : $\mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is optimized to predict node pairs $(u, v) \in \mathcal{V} \times \mathcal{V}$ generated from co-occurrences in unbiased or biased random walks. Many other embedding methods fit this *encoder-decoder* framework [19]: for example, factorization-based embeddings [20], [21] and even deep neural networks methods like [22], where the encoder function is given by a graph convolutional network [23].

B. Related Work

Interpretability for Node Embeddings: From the node embeddings perspective, interpretability is a multi-faceted concept that has been studied from different angles. In [3], [24] authors investigate, using prediction tasks, whether specific topological graph features (e.g., degree centrality, clustering coefficient, etc.) are encoded into node representations. These works significantly differ from our approach, where the aim is to find the comprehensible meaning of embedding dimensions, associating single dimensions with interpretable graph structures (e.g., communities). Other methods focus on measuring the interpretability of node embeddings with respect to node labels [4] and node centralities [25]. In [2] global interpretations are given as a hierarchy of graph partitions, but they do not focus on interpreting single dimensions. In [26] the authors study the impact on learned node embeddings when removing edges from the input graph. Instead, [27] estimates the importance of candidate nodes in each node representation. Another line of research focuses on producing interpretable-by-design representations based on graph clustering [28], [29], which are conceptually analogous to community-preserving node embeddings [30].

Interpretability for Link Prediction: Our approach focuses on the interpretation of embedding dimensions according to the graph structural reconstruction task, and it is related to methods for the interpretability of embedding-based link prediction. For instance, ExplaiNE [31] quantifies the variation in the probability of a link when adding or removing neighboring edges. PaGE-Link [32] generates explanations as paths connecting a node pair, while ConPI [33] provides the most influential interactions computed with an attention mechanism over the contextual neighborhoods. Other relevant methods study the problem of explaining link prediction in knowledge graphs [34], [35]. We, on the other hand, aim to explain the node embedding itself by associating explanations with each of its dimensions.

Interpretability for Word Embeddings: Since many methods for graph representation learning are based on language models, techniques for interpreting the dimensions of word embeddings are also relevant for node embeddings. Previous literature in this area focuses on interpreting the dimensions of word embeddings based on semantic information [36], [37], or analyzing geometric properties of the embedding space [38], [39]. Several other studies also propose approaches to learn interpretable representations by design, where the goal is achieving *sparsity* [40], [41], [42]. However, due to the high popularity of some embedding approaches, post-processing techniques that are built upon these approaches have been often preferred rather than interpretable-by-design methods [43], [44], [45].

III. EXPLAINING NODE EMBEDDINGS

We start by formalizing the desired fundamental properties of a global explanation for node embeddings. Intuitively, as graph structure serves as the input for generating unsupervised node embeddings, we seek reliable explanations in terms of associations between model parameters and human-understandable units of the input graph.

Decomposability: A global explanation should be able to refer to single parts of the model, and then explain these parts individually [46]. This is different from local instance-based explanation, where the focus is to interpret the result on single node predictions. In particular, a global explanation for node embeddings should be able to explain separately each dimension of the embedding space. To do so, in this work we extract interpretations in the form of important subgraphs $\mathcal{G}_d$ that we identify as the "meaning", or "explanation", of a dimension d.

Comprehensibility: An explanation should be human-understandable, in the sense that it relates to meaningful graph features [3] or discernible concepts [12]. With subgraph-based explanations, such features can be seen as structural components that we identify with the *communities* of the graph. For instance in biological networks like protein-protein interaction networks, these subgraphs could be important pathways responsible for biological mechanisms associated with for example a protein function or disease progression. In other graphs such as social networks these subgraphs can be seen as communities. Communities are typically considered as one of the fundamental organizing principles in these graphs [15] justifying their choice to identify the meaning of representation dimensions.

Sparsity: Explanations should be associated only with a minimal set of graph elements that sufficiently explain the learned parameters, ignoring the irrelevant ones [40], [41]. In our case, sparsity quantifies the spatial localization of an explanation subgraph.

Having defined the desired properties, we next describe how to obtain such decomposable explanations for node embeddings. In Section III-B we propose new metrics to quantify both the comprehensibility and the sparsity of these explanations.

A. Decomposable Explanations

Here we describe how we obtain global and decomposable explanations of node embeddings by extracting one explanation for every dimension of the latent space. Intuitively,

Algorithm 1: EXPLANATIONSUBGRAPH $(\mathcal{G}, \mathbf{X}(\mathcal{G}), d)$.

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Embedding matrix $\mathbf{X}(\mathcal{G}) \in \mathbb{R}^{D \times |\mathcal{V}|}$
Dimension to explain $d \in \mathcal{D} = \{1 \dots D\}$
Output: Explanation subgraph $\mathcal{G}_d$

Init. edge explanation set: $\mathcal{E}_d \leftarrow \emptyset$;
for $(u, v) \in \mathcal{E}$ **do**
  $\mathbf{u} \leftarrow \mathbf{X}_{:,u}$;
  $\mathbf{v} \leftarrow \mathbf{X}_{:,v}$;
  Compute edge utility:
  $\mu_d(\mathbf{u}, \mathbf{v}) \leftarrow \Delta_{\mathcal{D}}(\mathbf{u}, \mathbf{v}) - \Delta_{\mathcal{D}\setminus\{d\}}(\mathbf{u}, \mathbf{v})$;
  if $\mu_d(\mathbf{u}, \mathbf{v}) > 0$ **then**
    Add to explanation set: $\mathcal{E}_d \leftarrow \mathcal{E}_d \cup \{(u, v)\}$;
Edge-induced subgraph: $\mathcal{G}_d = \mathcal{G}[\mathcal{E}_d]$;
return $\mathcal{G}_d$;

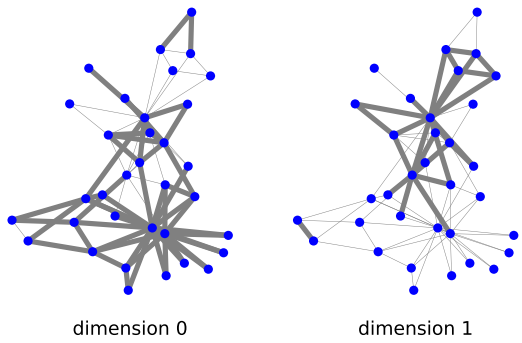


Fig. 2. Utility-induced subgraphs for 2-dimensional DEEPWALK embeddings trained on KARATE-CLUB.

given that embeddings are typically optimized for graph structure prediction, we aim to uncover the importance of individual dimensions in reconstructing the sub-structures of the graph. These substructures, consequently, will serve as explanations for individual dimensions. To extract the substructure explanations, we develop a utility function $\mu_d(\mathbf{u}, \mathbf{v})$ which quantifies the dimension's contribution in reconstructing a single graph edge with an embedding decoder. In fact, the score returned by the decoder $\text{dec}(\mathbf{u}, \mathbf{v})$ can be used to perform edge reconstruction, i.e., assessing the existence of edges (u, v): the higher the score, the higher the likelihood of observing the link on the input graph.

Here we adopt a simple yet effective approach for attributing dimension importance based on feature removal [47], [48]. Specifically, we define the attribution score of a single dimension $d \in \mathcal{D}$ in the reconstruction of an edge (u, v) as:

$$\mu_d(\mathbf{u}, \mathbf{v}) = \Delta_{\mathcal{D}}(\mathbf{u}, \mathbf{v}) - \Delta_{\mathcal{D}\setminus\{d\}}(\mathbf{u}, \mathbf{v}), \tag{2}$$

where $\Delta_{\mathcal{S}} : \mathbb{R}^{|\mathcal{D}|} \times \mathbb{R}^{|\mathcal{D}|} \to \mathbb{R}$ quantifies the average edge scoring of dimensions in the subset $\mathcal{S} \subseteq \mathcal{D}$

$$\Delta_{\mathcal{S}}(\mathbf{u}, \mathbf{v}) = \frac{1}{|\mathcal{S}|} \sum_{d \in \mathcal{S}} \mathrm{u}_d \mathrm{v}_d. \tag{3}$$

Notably, we consider a product-based scoring function that is appropriate to work with popular methods such as DEEPWALK and NODE2VEC. For an individual edge, the function in (2) measures how much the average likelihood increases or decreases when removing dimension d from the whole set $\mathcal{D}$. From a game-theoretic point of view, the importance scores $\mu_d(\mathbf{u}, \mathbf{v})$ defined above is an example of *marginal utility*, which expresses the contribution of dimension d when it is added to the *coalitional set* $\mathcal{S} = \mathcal{D} \setminus \{d\}$. A more exhaustive computation takes into account the average marginal contribution according to any possible coalitional set $\mathcal{S} \subset \mathcal{D}$ and it is given by the Shapley value [14], [49], [50]:

$$\phi_d(\mathbf{u}, \mathbf{v}) = \sum_{\mathcal{S} \subseteq \mathcal{D}\setminus\{d\}} \frac{\binom{|\mathcal{D}|-1}{|\mathcal{S}|}^{-1}}{|\mathcal{D}|} \left[\Delta_{\mathcal{S}\cup\{d\}}(\mathbf{u}, \mathbf{v}) - \Delta_{\mathcal{S}}(\mathbf{u}, \mathbf{v})\right], \tag{4}$$

where the difference $\Delta_{\mathcal{S}\cup\{d\}}(\mathbf{u}, \mathbf{v}) - \Delta_{\mathcal{S}}(\mathbf{u}, \mathbf{v})$ corresponds to the marginal utility of adding d to the dimensions' coalition $\mathcal{S} \subset \mathcal{D}$. A detailed description of (4) is reported in Appendix VI, available online. Therefore, the importance score $\mu_d(\mathbf{u}, \mathbf{v})$ corresponds to the marginal utility given by (4) with respect to the maximal coalitions ($|\mathcal{S}| = |\mathcal{D}| - 1$).

Since the exact computation of (4) has exponential time complexity, several approximation methods have been proposed in the literature to address scalability issues [49], [50]. Additionally, most of the approximations assume independence among features [14], [51] and suffer from considering feature correlations [52]. Rather than introducing an approximation, the marginal utility adopted here helps to derive computationally feasible formulas: in fact, the computation of μ_d reduces the time complexity from the order of $2^{|\mathcal{D}|-1}|\mathcal{D}||\mathcal{E}|$ to $|\mathcal{D}||\mathcal{E}|$ when computed over all edges of the graph. Moreover, since mutual independence of features is not usually guaranteed for node embeddings, the simplification is due to express the effect of isolated dimensions disregarding possible feature correlations.

We use the importance scores defined in (2) to determine the explanation subgraphs formed by the edges that benefit most from the presence of a dimension d. Specifically, we identify the subgraph $\mathcal{G}_d \equiv \mathcal{G}[\mathcal{E}_d]$ induced by links $\mathcal{E}_d = \{(u, v) \in \mathcal{E} : \mu_d(\mathbf{u}, \mathbf{v}) > 0\}$ with *positive* marginal utility as the explanation of dimension d. We restrict ourselves to positive payoffs because the main interest is to find those dimensions which are more effective in predicting a given edge, leaving for future work the analysis of the negative effects. In Algorithm 1 we provide the pseudo-code for extracting explanation subgraphs starting from a pre-trained graph embedding. In Fig. 2 we highlight the explanation subgraphs in the KARATE-CLUB dataset for 2-dimensional DEEPWALK embeddings. We say that subgraphs depicted in Fig. 2 are global explanations of DEEPWALK because they allow associating any model dimension with pieces of the data, and provide a global interpretation that is decomposed into per-dimension views.

B. Measuring Comprehensibility and Sparsity

Here we define metrics to quantify the quality of the extracted subgraph explanations. Specifically, we introduce two interpretability metrics to measure the comprehensibility and sparsity of the per-dimension induced subgraphs.

Community-Aware Metric: Let $\mathcal{E}_d$ denote the set of edges in the explanation subgraph for dimension d. Given the

information on important subgraphs for example pathways in case of biological networks or communities in social networks, we measure the relevance of explanation subgraphs to these communities/subgraphs using precision and recall scores. Let $\mathcal{P}(\mathcal{G}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ denote the set of ground-truth link partitions/communities/subgraphs of the input graph. Later in the experiment section, we will also describe how to obtain such ground-truth subgraphs when these are not given. We first compute precision and recall metrics, which measure the association strength of extracted explanation subgraphs with the given ground-truth important subgraphs/communities.

$$\text{precision}\,(\mathcal{E}_d, \mathcal{P}_i) = \frac{|\mathcal{E}_d \cap \mathcal{P}_i|}{|\mathcal{E}_d|}, \quad \text{recall}(\mathcal{E}_d, \mathcal{P}_i) = \frac{|\mathcal{E}_d \cap \mathcal{P}_i|}{|\mathcal{P}_i|} \tag{5}$$

We then compute interpretability score I_d as the maximum F1-score over all given ground-truth communities:

$$I_d^{(com)} = \max_{\mathcal{P}_i \in \mathcal{P}(\mathcal{G})} \text{F1}(\mathcal{E}_d, \mathcal{P}_i) \tag{6}$$

where F1-score is the harmonic mean between precision$(\mathcal{E}_d, \mathcal{P}_i)$ and recall$(\mathcal{E}_d, \mathcal{P}_i)$. Higher values of I_d indicate the dimension d is strongly associated with a single community. Global community-aware interpretability can be quantified with the average $I^{(com)} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} I_d^{(com)}$.

Sparsity-Aware Metric: In the absence of ground-truth community information, we can anyhow quantify in an unsupervised manner whether dimensions can highlight structure-relevant subgraphs. In particular, without any cognition on community structure, it is highly preferable that interpretable directions of the embedding space are associated with a minimal set of significant edges. Inspired by explanation masks in graph neural networks [53], we formulate its calculation using Shannon entropy [9]:

$$I_d^{(sp)} = -\frac{1}{\log |\mathcal{E}|} \sum_{(u,v) \in \mathcal{E}} \left(\frac{[(u,v) \in \mathcal{E}_d]}{z_d} \right) \log \left(\frac{[(u,v) \in \mathcal{E}_d]}{z_d} \right) \tag{7}$$

where the function $[*]$ returns 1 if the proposition inside is true (and 0 otherwise), and $z_d = \sum_{(u,v) \in \mathcal{E}} [(u,v) \in \mathcal{E}_d]$ is a normalization for the correct computation of the Shannon entropy. Lower values indicate that embedding dimensions are associated with smaller-sized subgraphs. Global sparsity-aware interpretability can be quantified with the average $I^{(sp)} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} I_d^{(sp)}$.

IV. Our Approach: Dimension-Based Interpretable Node Embedding

In previous sections, we proposed the utility-induced subgraphs as explanations to interpret node embedding dimensions. Unsurprisingly, as we show in Fig. 1(a) for DeepWalk, typically it is difficult to map utility-induced subgraphs to interpretable graph units, mainly because these methods are trained with the unique goal of maximizing reconstruction performance. Filling this gap, we introduce *Dimension-based Interpretable Node Embedding* (Dine), a novel method to improve the interpretability of already trained node embeddings by retrofitting the induced subgraphs which affect interpretation metrics (Section III-B).

We design such retrofit task with an autoencoder architecture, trained to reconstruct embedding vectors in the input [54]. By encoding the input node representations into a hidden feature space, the autoencoder can be regularized in order to promote the learning of interpretable dimensions. Despite the many existing regularizations already used to obtain interpretable embeddings, such as non-negativity [42], [55] or sparsity [41], [43], in this work we employ *orthogonality* regularizers [56], [57], [58] to achieve the purpose. Orthogonality is closely related with *disentaglement* [59], [60], which is a key concept implemented in several methods for decoupling correlations between latent dimensions [61], [62], with the results of learning more compact representations whose feature dimensions are associated with independent facets of data.

We argue improving dimension-based orthogonality is more effective for several reasons:

- *Distinct features:* If two dimensions are orthogonal, it means that they are independent of each other and do not share any latent factor. Therefore, each dimension provides unique information which can be interpreted as representing a distinct characteristic of the data.
- *Separation of concepts:* Orthogonal dimensions in the embedding space can represent independent concepts. For example, in the context of word embeddings, the concept of "gender" might be captured along one dimension, while the concept of "age" might be captured along another. This helps us to easily separate and understand these different features of the data.
- *Removing redundancy:* Orthogonality implies no redundancy. If two dimensions are not orthogonal, then they project onto each other to some extent, meaning there's some shared information. This shared information could be interpreted as redundancy. By ensuring orthogonality, we ensure that each latent direction provides new, unique information.
- *Clear interpretation of distances:* In an orthogonal space, distances directly correlate with dissimilarity. For instance, two orthogonal word embeddings would likely represent words with very different meanings or features, while vectors closer together would represent more similar words.

Contrary to previous works [58] that enforce orthogonality of neural weights, here we employ orthogonalization of the edge reconstruction patterns that directly affect per-dimension utility subgraphs. In this way, we obtain node embeddings whose interpretability is optimized according to the metrics introduced in the previous section. In the next, we first show how we can rephrase the utility optimization in an effective way to be easily handled, how the method is implemented and the motivations behind its functioning.

A. Optimization of Marginal Utilities

Starting with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an embedding encoder enc : $\mathcal{V} \to \mathbb{R}^D$, Dine aims to learn an opportune mapping $h : \mathbb{R}^D \to \mathbb{R}^K$ (both $D, K << |\mathcal{V}|$) in such a way the output

per-dimension subgraphs are highly interpretable in terms of decomposability, comprehensibility and sparsity of explanations. We use $h(\mathbf{v}) = (h \circ \text{enc})(v)$ to indicate the embedding vectors of node $v \in \mathcal{V}$ mapped with a standard encoding function and equipped with the interpretable layer h. Resulting embedding vectors are collected into the matrix $\mathbf{H} \in \mathbb{R}^{K \times |\mathcal{V}|}$, such that $\mathbf{H}_{:,v} = h(\mathbf{v})$. We also refer to K as the cardinality of the set containing the enumerated dimensions of the new space $\mathcal{K} = \{1 \dots K\}$.

Since subgraphs are the results of positive marginal utilities, we are interested in optimizing the utility measures $\mu_d(h(\mathbf{u}), h(\mathbf{v}))$ as a function of the new embedding parameters defined by h. In the following paragraphs we show that, assuming the new embedding space to be the unit-size hypercube $[0,1]^K \subset \mathbb{R}^K$, together with sufficiently high embedding dimensionality, we can simplify the optimization of the utility measure. In fact in the following theorem we show that for a given edge, the interpretability (utility) measure for a dimension can be approximated using a single dot product over the embedding pair.

Theorem 1: Let be $\mathbf{H} \in [0,1]^{K \times |\mathcal{V}|}$ a node embedding matrix of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in the K-dimensional hypercube. For high dimensionality K, the per-dimension utility score for edge $(u, v) \in \mathcal{E}$, $\mu_d(h(\mathbf{u}), h(\mathbf{v}))$, can be expressed as:

$$\mu_d\left(h(\mathbf{u}), h(\mathbf{v})\right) = \frac{H_{d,u} H_{d,v}}{K} + \mathcal{O}\left(\frac{1}{K^2}\right) \tag{8}$$

The proof is given in Section I of the Appendix, available online.

We define $H_{d,u} H_{d,v} / K$ as the entries $M_d(u, v)$ of K continuous-valued graph masks $\{\mathbf{M}_d \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}\}_{d \in \mathcal{K}}$, formally derived from the outer products of the rows of $\mathbf{H}$, i.e., $\mathbf{M}_d = \mathbf{H}_{d,:} \otimes \mathbf{H}_{d,:}$. The (8) tells that $M_d(u, v)$ and $\mu_d(h(\mathbf{u}), h(\mathbf{v}))$ differ by a negligible term $\mathcal{O}(\frac{1}{K^2})$. Consequently, we can optimize the quantities $M_d(u, v)$ that are computed from individual products. Graph masks have the role of highlighting the structure-relevant edges for any direction in the K-dimensional learned space.

B. Method Implementation

We now describe the retro-fitting optimization task and identify the key design choices which allow obtaining more interpretable induced subgraphs, and whose effectiveness is shown in the experiments section. We provide an in-depth discussion on the motivations behind the algorithm design in Section IV-C. Please refer to Fig. 3 for a schematic diagram of our approach.

We implemented h as the latent projection of a single-layer autoencoder, namely $\mathbf{H} = \sigma(\mathbf{W}^{(0)}\mathbf{X} + \mathbf{b}^{(0)})$, which returns $\tilde{\mathbf{X}} = \mathbf{W}^{(1)}\mathbf{H} + \mathbf{b}^{(1)}$ as output. The hidden layer matrix of the autoencoder, $\mathbf{H} \in \mathbb{R}^{K \times |\mathcal{V}|}$, collects the interpretable embedding vectors that we aim to learn. We add regularization constraints on the hidden embedding matrix $\mathbf{H}$ while training the autoencoder, in order to learn optimal graph masks. Specifically, we minimize the following loss:

$$\mathcal{L} = \mathcal{L}_{ac}\left(\mathbf{X}, \tilde{\mathbf{X}}\right) + \sum \mathcal{L}_{reg}(\mathbf{H}) \tag{9}$$

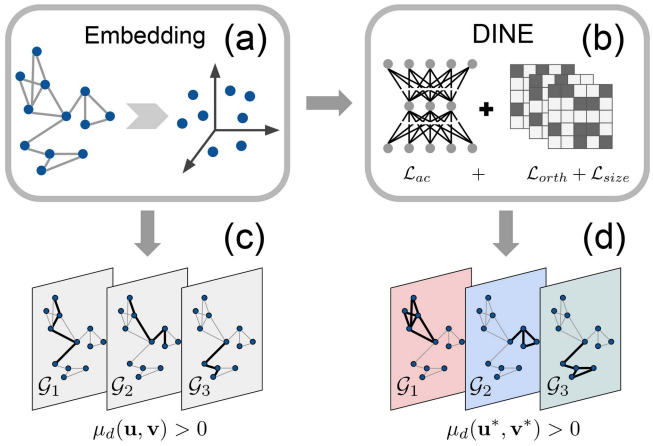


Fig. 3. Schematic view of our methodology. Starting from a graph embedding representation (a), we apply the method DINE (b). Explanations are given in the form of per-dimension subgraphs, both for the starting embedding (c) and the DINE embedding (d).

where $\mathcal{L}_{ac}(\mathbf{X}, \tilde{\mathbf{X}}) = ||\mathbf{X} - \tilde{\mathbf{X}}||_F$ is the reconstruction error between the input and output embedding matrices $\mathbf{X}, \tilde{\mathbf{X}} \in \mathbb{R}^{D \times |\mathcal{V}|}$. We jointly optimise masks matrices $\{\mathbf{M}_d \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}\}_{d \in \mathcal{K}}$, computed in function of hidden layer weights matrix $\mathbf{H}$, and the autoencoder parameters $\{\mathbf{W}^{(0)} \in \mathbb{R}^{K \times D}, \mathbf{W}^{(1)} \in \mathbb{R}^{D \times K}, \mathbf{b}^{(0)} \in \mathbb{R}^K, \mathbf{b}^{(1)} \in \mathbb{R}^D\}$. We determined the following regularization terms as optimal for promoting interpretable dimensions:

- Induced subgraphs might have minimal overlaps between each other in order to be interpreted as communities. Inspired by graph clustering [63], we squeeze embedding mask matrices into one partition matrix $\mathbf{P} \in \mathbb{R}^{K \times |\mathcal{V}|}$, with entries $P_{d,v} = \sum_u M_d(u, v)$ computed by aggregating edge reconstruction scores with the same target node.[1] To encourage relevant subgraphs to be incorporated into different embedding axes, we optimize the following *Orthogonality Loss:*

$$\mathcal{L}_{orth}(\mathbf{P}) = \left|\left| \frac{\mathbf{P}\mathbf{P}^{\mathrm{T}}}{||\mathbf{P}\mathbf{P}^{\mathrm{T}}||_F} - \frac{\mathbf{1}_K}{||\mathbf{1}_K||_F} \right|\right|_F \tag{10}$$

- To avoid degenerate solutions due to the orthogonality constraint, e.g., all relevant subgraphs reconstructed in the same dimension, we enforce the size of every mask $\mathrm{s}_d = \sum_{u,v} M_d(u, v)$ to be non-zero. This constraint is accomplished by maximizing the entropy of the vector containing size variables $\mathbf{s}$, or equivalently minimizing the *Size Loss:*

$$\mathcal{L}_{size}(\mathbf{s}) = \log K + \sum_{d \in \mathcal{K}} \left(\frac{\mathrm{s}_d}{\sum_{q \in \mathcal{K}} \mathrm{s}_q}\right) \log \left(\frac{\mathrm{s}_d}{\sum_{q \in \mathcal{K}} \mathrm{s}_q}\right) \tag{11}$$

The use of node-based partitions is due to scalability reasons: the entries of the partition matrix $P_{d,v} = \sum_u M_d(u, v) \propto H_{d,v}[\sum_u H_{d,u}]$ can be computed just by multiplying the hidden matrix elements $H_{d,v}$ by the quantities $\mathrm{f}_d = \sum_u H_{d,u}$ collected

[1] Aggregating over the source node would give the same result since we work with undirected graphs and $M_d(u, v) = M_d(v, u)$.

in vector $\mathbf{f}$, avoiding the explicit calculation of graph masks with outer products, then reducing the complexity from $\mathcal{O}(K \times |\mathcal{V}|^2)$ to $\mathcal{O}(K \times |\mathcal{V}|)$. The full objective loss is given by[2]:

$$\mathcal{L} = \mathcal{L}_{ac}\left(\mathbf{X}, \tilde{\mathbf{X}}\right) + \mathcal{L}_{orth}(\mathbf{P}) + \mathcal{L}_{size}(\mathbf{s}) \tag{12}$$

In Section III of the Appendix, available online, we include the pseudo-code for DINE and related time complexity analysis in comparison with other embedding methods also used in the experiments.

C. Discussion on the Regularization Losses

For simplicity we consider binary edge masks $\tilde{M}_d(u,v) \in \{0,1\}$. Nevertheless, when training our model with gradient-based optimizers, we recast the problem into a continuous formulation by learning the aforementioned real-valued masks[3] $M_d(u,v) \in \mathbb{R}$. The mask optimization consists of clustering the input graph by discovering per-dimension subgraphs, i.e., selecting edges (u,v) based on their importance (or irrelevance) for the explanation of dimension d:

$$\tilde{M}_d(u,v) = [(u,v) \in \mathcal{E}_d]$$

Toward that, we have defined a node-level partition matrix $\tilde{\mathbf{P}} \in \mathbb{R}^{K \times |\mathcal{V}|}$ which, in the binary masks case, takes the form:

$$\tilde{P}_{d,v} = \sum_{u \in \mathcal{V}} \tilde{M}_d(u,v) = \deg_{\mathcal{G}_d}(v) \tag{13}$$

where $\deg_{\mathcal{G}_d}(v)$ is the degree of node v restricted to the subgraph $\mathcal{G}_d$. Inspired by spectral graph clustering [63], in (10) we have proposed $\mathcal{L}_{orth}$ to learn explanation subgraphs by constraining the rows of the assignment matrix to be orthogonal. In fact, by observing the off-diagonal entries of the matrix $\tilde{\mathbf{P}}\tilde{\mathbf{P}}^{\mathrm{T}} \in \mathbb{R}^{K \times K}$:

$$\tilde{P}\tilde{P}^{\mathrm{T}}_{d,q} = \sum_{v \in \mathcal{V}} \deg_{\mathcal{G}_d}(v) \cdot \deg_{\mathcal{G}_q}(v)$$

it can be noticed that pushing them to zero will force nodes to have connections with other vertices only in a specific dimensional subgraph (and zero-degree elsewhere), discouraging overlaps between subgraphs.

Constraining diagonal entries of the orthogonalizer $\tilde{P}\tilde{P}^{\mathrm{T}}_{d,d} = \sum_v [\deg_{\mathcal{G}_d}(v)]^2$ to be greater than zero, while ensuring the discovery of densely connected subgraphs, does not prevent gradient-based optimization to find degenerate clustering solutions. As pointed out in [63], [65], passing to a continuous relaxation of node assignments conceivably leads to trivial or sub-optimal clusters.[4] In (11) we have proposed an additional regularization term to avoid this issue. With $\mathcal{L}_{size}$ we require the subgraph sizes $\tilde{\mathrm{s}}_d = \sum_v \deg_{\mathcal{G}_d}(v)$ to be stable across dimensions, with a maximum entropy formulation.

[2]There is no discrepancy between the following definition and (9), because both $\mathbf{P}$ and $\mathbf{s}$ are functions of hidden matrix $\mathbf{H}$.

[3]Also, the binary mask constraint would result in learning sparse and binary vectors, with beneficial results only by hugely increasing the embedding size [43], [64], which is not the purpose of this work.

[4]The work [63] focuses on the continuous relaxation of a binary assignment matrix $\tilde{\mathbf{C}} \in \{0,1\}^{K \times |\mathcal{V}|}$, but we found similar issues in our case with "degree-based" partition matrix $\tilde{\mathbf{P}}$.

TABLE I
SUMMARY STATISTICS ABOUT REAL-WORLD GRAPH DATA

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\ \rvert$	$\lvert\mathcal{C}(\mathcal{G})\rvert$	# classes	NMI
CORA	2,485	5,069	28	7	0.452
CITESEER	2,110	3,668	35	6	0.319
PUBMED	19,717	44,324	38	3	0.201
BLOG	5,196	171,743	10	6	0.231
FLICKR	7,575	239,738	9	9	0.158
WIKI	2,357	11,592	17	17	0.387

In order: number of nodes $|\mathcal{V}|$, number of edges $|\mathcal{E}|$, number of extracted communities with Louvain method $|\mathcal{C}(\mathcal{G})|$, number of attached node classes, and Normalized mutual information between the two set of labels.

V. EXPERIMENTS

In this section, we present the results of our study on the DINE model from different perspectives. The main objective is to address the following research questions:

(RQ1) How does the interpretability of DINE compare to those of standard embedding techniques?
(RQ2) How well does DINE perform over graph downstream tasks, such as link prediction and node classification?
(RQ3) Is DINE suitable for practical use, particularly in scenarios requiring scalability?

In the following sections, we describe the data, models, and tasks used in the comparison to address our research questions.

A. Data and Models

We present our results on a variety of benchmark datasets used in prior work [66]: three citation networks (CORA, CITESEER and PUBMED), two social networks (BLOG and FLICKR), and a web pages network (WIKI). Despite their original format, we restrict our analysis to the largest connected component of any graph, considered unweighted and undirected. As described in Section III-B, we rely on *ground-truth link partitions* for interpretability metrics based on community structure. Many empirical graphs have node metadata that can be used for node-to-community mapping. However, the use of metadata as structure-aware labels has recently been criticized by previous works [67], [68]. Instead, we use community detection to discover partition labels. We avoid computationally expensive and overlapping community detection methods [69], [70], [71], [72] and use the arguably intuitive and simpler Louvain detection method [73] to derive edge labels based on node-level graph communities. Specifically, we run Louvain detection method [73] to extract the node-level communities $\mathcal{C}(\mathcal{G}) = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ and we assign partition label for a given edge (u,v) the set $\{c(u), c(v)\}$, where $c : \mathcal{V} \to \mathcal{C}(\mathcal{G})$ is the node-level community membership function. We report dataset statistics in Table I, where we also show that Louvain labels and node classes are highly uncorrelated, justifying the choice of using community detection to extract structure-aware labels.

We consider as baseline methods different dense and sparse embedding approaches: DEEPWALK [17], INFWALK [74], GAE [22], Modularity-aware GAE (MODGAE) [75], GEMSEC [29], and SPINE [43]. Details about hyper-parameters and

training settings used in each method can be found in Section II of the Appendix, available online.

B. Tasks Description

For answering *RQ1*, we measure the interpretability of DINE in comparison to our baseline methods. To do so, we compute for any embedding dimension interpretability scores that we have defined in Section III-B. Instead of averaging over all dimensions for comparing the models, we focus on a subset of effective dimensions $\mathcal{D}_{eff}$ that encode the majority of edge information, to avoid potential noise from the less important dimensions. Specifically, after computing $I_d^{(com)}$ or $I_d^{(sp)}$, we select the top-ranked dimensions that cumulatively contribute to the reconstruction of at least 90% of the graph edges, i.e., $|\bigcup_{d\in\mathcal{D}_{eff}} \mathcal{E}_d| = 90\%|\mathcal{E}|$. Thus we compute global scores as $I_{eff}^{(com|sp)} = \frac{1}{|\mathcal{D}_{eff}|}\sum_{d\in\mathcal{D}_{eff}} I_d^{(com|sp)}$.

For answering *RQ2*, we measure link prediction and node classification performances of DINE in comparison to baseline methods. For link prediction, before training every method, we randomly remove 10% of the edges that are used as positive examples for the task. We also sample the same number of node pairs from the set of non-existing links as negative examples. The task consists in ranking the collected node pairs with the product-based edge decoder function and evaluating the classification performance with the ROC-AUC score. For node classification, we train a multi-label logistic regression over a random split of 80% of the nodes, and we evaluate the F1 scores of the classification performance by predicting the class labels attached to nodes of the remaining 20%.

For answering *RQ3*, we measure the training time of DINE in comparison to baseline methods.

C. Results

In our experiments, all methods, with the exception of SPINE, are trained to produce embedding vectors with sizes K in the set $\{2, 4, 8, 16, 32, 64, 128\}$, referred to as *output dimensions*. On the other hand, the dimensionalities D of vectors used for training DINE and SPINE, referred to as *input dimensions*, are taken from the set $\{8, 16, 32, 64, 128, 256, 512\}$. For SPINE, due to the presence of the overcomplete layer, we chose output dimensions to be multiples of the input dimensions, i.e., $K = \tau D$ with τ between $\times 1$ and $\times 8$.

In our comparison, we evaluated DINE against both dense and sparse methods. For the comparison with dense methods, DINE was trained using DEEPWALK and GAE vectors, with performance reported in the figures across different output dimensions, choosing the best score among the input dimensions. To compare with sparse methods, both SPINE and DINE were trained using DEEPWALK vectors, with their performance reported in the figures across different input DEEPWALK, choosing the best score among the output dimensions. The overall best results for dense and sparse embeddings are reported respectively at the top and at the bottom of each table, with average evaluation score and standard deviation computed over 5 separate training runs.

TABLE II
COMMUNITY-AWARE SCORES FOR DIFFERENT EMBEDDING METHODS

	CORA	CITESEER	PUBMED	BLOG	FLICKR	WIKI
DEEPWALK	44.8±0.8	43.3±1.4	42.2±1.1	43.2±0.7	49.9±10.5	44.5±2.5
INFWALK	50.4±0.0	50.9±0.0	39.1±0.0	49.7±0.0	41.0±0.0	43.3±0.0
GAE	37.2±0.6	42.0±1.0	48.7±1.9	49.6±0.6	62.5±2.1	46.0±1.4
MODGAE	56.7±3.6	45.0±2.1	52.3±1.8	48.4±0.8	64.4±2.0	43.4±3.9
GEMSEC	40.5±1.3	44.6±1.4	43.1±0.9	39.2±3.0	42.1±2.2	45.4±0.2
DW+DINE	**62.3±1.9**	**64.1±2.7**	**60.5±2.7**	59.0±2.3	**65.7±1.4**	**65.2±1.0**
GAE+DINE	55.0±1.5	52.6±0.8	59.3±1.1	**60.0±0.8**	59.5±1.3	63.1±1.7
DEEPWALK	49.5±2.3	49.1±1.6	45.7±0.6	48.3±0.9	46.9±1.8	49.6±0.9
DW+SPINE	59.0±5.1	54.3±6.0	**60.8±3.7**	**63.2±8.2**	**70.3±0.1**	61.1±1.5
DW+DINE	**62.3±1.9**	**64.1±2.7**	60.5±2.7	59.0±2.3	65.7±1.4	**65.2±1.0**

We highlight the best (highest) score.

TABLE III
SPARSITY-AWARE SCORES FOR DIFFERENT EMBEDDING METHODS

	CORA	CITESEER	PUBMED	BLOG	FLICKR	WIKI
DEEPWALK	79.2±0.6	77.8±0.6	84.9±0.1	86.5±0.2	87.4±0.4	82.0±0.3
INFWALK	69.6±0.0	**62.4±0.0**	77.4±0.0	84.3±0.0	87.0±0.0	76.0±0.0
GAE	83.7±0.5	82.5±0.4	85.1±0.5	87.1±0.9	78.5±0.1	84.4±0.2
MODGAE	73.9±1.9	73.4±7.7	85.1±0.7	82.9±7.5	60.0±2.9	72.6±1.4
GEMSEC	82.7±0.3	81.7±0.3	86.5±0.2	91.1±0.3	91.6±0.2	84.6±0.4
DW+DINE	**66.0±1.2**	63.0±1.5	**68.4±24.2**	**74.9±0.6**	**37.2±6.9**	**68.8±0.3**
GAE+DINE	72.8±0.1	72.8±0.9	80.5±0.3	82.0±1.2	71.1±2.1	75.6±0.3
DEEPWALK	76.7±0.5	75.5±0.3	83.6±0.1	84.0±0.1	87.4±0.4	79.6±0.4
DW+SPINE	69.7±2.0	70.3±2.5	**74.1±3.3**	**61.0±8.9**	72.9±12.5	**52.9±9.7**
DW+DINE	**66.0±1.2**	**63.0±1.5**	74.8±0.8	74.9±0.6	**68.1±6.0**	68.8±0.3

We highlight the best (lowest) score.

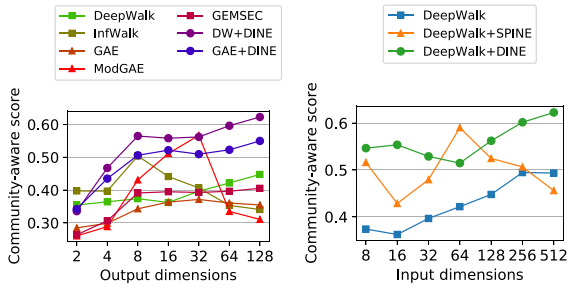


Fig. 4. Community-aware scores for CORA (higher is better). On the left we compare DINE with dense embedding methods; on the right, we compare DINE with sparse embedding SPINE.

In the Appendix, available online, additional figures are reported in Section IV with all the results not shown in the main paper. Moreover, in Section V we report supplementary experiments on the interpretability performances, such as ablation studies for the removal of individual regularization terms in DINE, and robustness of the metrics w.r.t. noise and random walk hyperparameters of the input embeddings.

Interpretability (RQ1): Best scores for interpretability metrics $I_{eff}^{(com)}$ and $I_{eff}^{(sp)}$ are reported in Tables II and III respectively for all the datasets, with detailed plots in Figs. 4 and 5 for CORA. On the top of each table, showing the comparison with dense embeddings, we notice that the combination DEEPWALK+DINE performs well in almost every dataset. Moreover, we observe that GAE+DINE is less interpretable than DEEPWALK+DINE, but still more interpretable than the other dense baselines. From the comparison with sparse embeddings, on the bottom

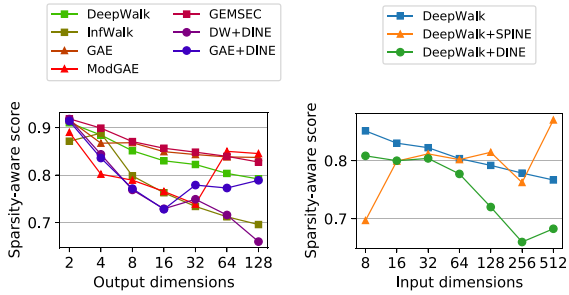


Fig. 5. Sparsity-aware scores for CORA (lower is better). On the left we compare DINE with dense embedding methods; on the right, we compare DINE with sparse embedding SPINE.

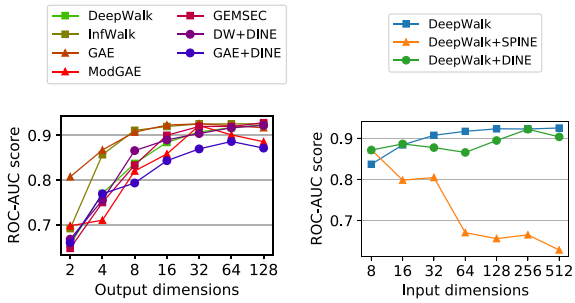


Fig. 6. ROC-AUC scores for link prediction in CORA. On the left we compare DINE with dense embedding methods; on the right, we compare DINE with sparse embedding SPINE.

TABLE IV
ROC-AUC SCORES FOR DIFFERENT EMBEDDING METHODS

	CORA	CITESEER	PUBMED	BLOG	FLICKR	WIKI
DEEPWALK	92.3±0.7	94.5±0.7	**96.4±0.2**	75.6±0.1	65.6±0.1	84.8±0.3
INFWALK	92.6±0.8	94.3±0.8	93.9±0.2	67.7±0.4	62.6±0.2	88.6±0.2
GAE	92.6±0.4	**95.2±0.6**	95.8±0.2	86.9±0.3	89.4±0.4	**94.2±0.2**
MODGAE	92.2±1.0	94.0±0.9	95.1±0.2	**88.0±0.2**	90.6±0.6	92.8±0.4
GEMSEC	**92.8±0.6**	94.2±0.7	95.1±0.3	79.4±0.2	67.0±0.3	90.5±0.4
DW+DINE	92.2±0.7	94.7±1.0	93.3±1.2	74.7±0.6	75.9±0.9	90.5±0.4
GAE+DINE	88.6±1.2	92.0±0.9	96.1±0.1	82.0±1.0	**91.1±0.3**	**94.2±0.4**
DEEPWALK	**92.5±0.6**	**94.9±0.8**	**96.5±0.1**	**75.8±0.1**	66.7±0.2	84.8±0.2
DW+SPINE	87.2±1.6	89.7±2.1	90.2±0.6	67.3±1.4	63.2±3.5	81.6±1.4
DW+DINE	92.2±0.7	94.7±1.0	93.3±1.2	74.7±0.6	**75.9±0.9**	**90.5±0.4**

We show in boldface letters the best score, and we highlight in gray the methods that reach at least 95% of the best score.

TABLE V
MICRO-F1 SCORES FOR DIFFERENT EMBEDDING METHODS

	CORA	CITESEER	PUBMED	BLOG	FLICKR	WIKI
DEEPWALK	83.4±0.7	73.3±2.3	80.9±0.5	65.7±1.5	53.2±1.0	64.1±1.6
INFWALK	81.6±1.4	73.2±0.8	**81.3±0.6**	65.6±0.5	**59.3±0.8**	64.4±1.7
GAE	84.0±1.9	70.5±2.7	75.9±0.5	71.2±0.9	37.5±1.3	**69.7±1.8**
MODGAE	84.3±1.5	72.0±1.3	79.7±0.8	**72.1±1.0**	41.1±1.5	68.2±1.3
GEMSEC	**85.6±1.2**	**75.5±1.8**	79.1±1.1	65.2±1.0	45.7±0.9	69.0±1.3
DW+DINE	81.5±2.6	73.0±1.0	78.9±0.6	64.8±1.1	47.5±1.5	62.5±2.0
GAE+DINE	75.5±1.6	61.9±2.1	70.7±1.2	65.7±1.3	23.7±3.5	60.8±1.9
DEEPWALK	**84.2±0.7**	**74.2±3.3**	**82.1±0.9**	**65.7±1.5**	**53.2±1.0**	**64.1±1.6**
DW+SPINE	78.2±1.9	67.3±0.9	77.2±0.8	62.6±1.5	44.2±0.7	59.7±1.0
DW+DINE	81.5±2.6	73.0±1.0	78.9±0.6	64.8±1.1	47.5±1.5	62.5±2.0

We show in boldface letters the best score, and we highlight in gray the methods that reach at least 95% of the best score.

of each table, both DINE and SPINE trained on DEEPWALK demonstrate good interpretability, obtaining the best scores in half of the datasets each. Our results confirm the well-known property of vector sparsity improving the interpretability of representations [76]. On Figs. 4 and 5, we observe that both interpretability metrics improve as the embedding dimensions increase in CORA. This is true also in the other datasets shown in Section IV of the Appendix, available online, providing important guidance for choosing the appropriate embedding size in real-world applications.

Downstream Tasks (RQ2): ROC-AUC and Micro-F1 scores are documented in Tables IV and V, with a detailed illustration for CORA in Figs. 6 and 7. From the comparison of dense methods in downstream tasks, on the top of each table, results show that in citation networks DINE retrofitted embeddings

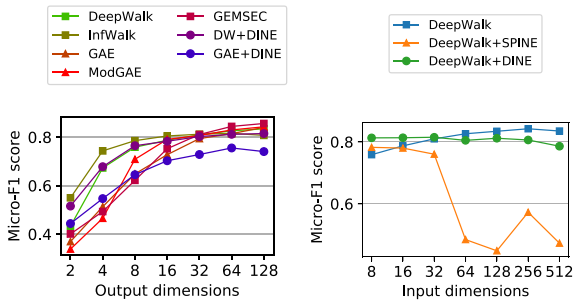


Fig. 7. Micro-F1 scores for node classification in CORA. On the left we compare DINE with dense embedding methods; on the right, we compare DINE with sparse embedding SPINE.

perform similarly to the best dense models. This result implies that we do not have to trade task performance with increased interpretability. In other datasets, the best link prediction scores are obtained by MODGAE and GAE+DINE; while for node classification the most successful models are MODGAE, INFWALK and GAE. Differently from link prediction, where the maximum score drop is 6% in GAE+DINE, our model has a performance drop greater than 10% in node classification outside citation networks: we suppose this is due to the misalignment between community structure and node classes, with DINE that fails to fit the second ones. In addition, when input models already have weak performances (e.g., DEEPWALK, or GAE in FLICKR), it is unlikely that DINE could obtain further improvements.

When comparing sparse embeddings in link prediction, DINE *consistently outperforms* SPINE, with comparable or even superior results (in the case of FLICKR and WIKI) to DEEPWALK. When comparing sparse embeddings in node classification, DINE is still superior to SPINE, but without reaching DEEPWALK scores. Interestingly, our results in Figs. 6 and 7 also demonstrate that SPINE's performance decreases with increasing input dimensions, unlike the other methods.

Scalability (RQ3): The training times for various methods are presented in Fig. 8, with the intervals normalized relative to the number of iterations/epochs. For DEEPWALK, the intervals are further divided with respect to the `num_walks` parameter to remove the dependence on the number of walks per node. The left panel shows that the runtimes for DEEPWALK, SPINE, and DINE increase with the number of nodes, while the center panel demonstrates that the execution time for GAE increases with the

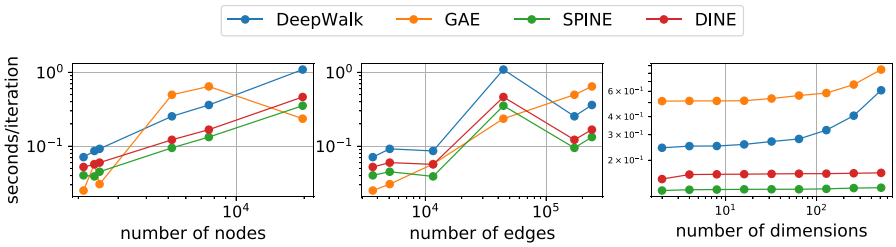


Fig. 8. Normalized run times for different embedding methods when trained on multiple datasets: in the left, run times for 128-dim output embeddings while varying the number of nodes; in the center, run times with 128-dim output embeddings while varying the number of edges; in the right, run times varying the number of embedding dimensions in FLICKR dataset.

number of edges. Additionally, DINE has slightly longer training times compared to SPINE, but both are faster than DEEPWALK. The right panel indicates that the training time for SPINE and DINE has a weak dependence on the number of embedding dimensions, while this dependence is more pronounced in GAE and DEEPWALK. Experimental results on the scalability suggest that it is possible to increase the interpretability of node representations without requiring significant computational costs. Additionally, the analysis of runtime complexities included in Section III of the Appendix, available online, confirms the empirical findings.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we presented a framework for constructing global explanations for node embeddings. We explain each embedding dimension using the important substructures of the input graph. To construct these explanations we developed a new *model-agnostic* utility measure which computes the contributions of each dimension to predict the graph structure. Our explanations follow the desired properties of Decomposability, Comprehensibility and Sparsity.

With the goal of maximizing these properties, we proposed and developed DINE, an auto-encoder framework to enhance the interpretability of existing node embeddings. In short, DINE captures the structural properties encoded in an input embedding and optimizes a set of graph masks in order to promote orthogonality and sparsity of predicted sub-structures. Our comprehensive experimental study supports our claims that DINE improves embedding comprehensibility over standard node embedding techniques, without compromising the task performance. In particular, combining DEEPWALK with DINE allows obtaining interpretable node embeddings with effective performances in downstream tasks. DINE is also preferable to the sparse method SPINE due to its better achievements in link prediction and node classification. DINE scales well with respect to the input graph size, being suitable to be used in graphs with high edge density. Since the computation of the exact utility measure has exponential complexity as is usually the case for Shapley-based measures, the presented utility measure shares limitations common to other approximation strategies suggested in the literature. In particular, the approximation deteriorates under high interdependence among embedding features [52]. Nevertheless, the encouraging results from our experiments support the effectiveness of this approach.

These contributions open multiple avenues for future work. Specifically, our approach can be extended to constructing interpretable node embeddings whose dimensions are aware of multi-scale subgraph structures [77] inherent in many real-world graphs [78]. DINE can also be used as a plug-in architecture to facilitate interpretable learning in various graph neural network encoders [22]. Furthermore, future investigations could explore the impact on representations learned in supervised or semi-supervised settings [79].

REFERENCES

[1] W. L. Hamilton, *Graph Representation Learning*, vol. 14. San Rafael, CA, USA: Morgan & Claypool, 2020, pp. 1–159.
[2] N. Liu, X. Huang, J. Li, and X. Hu, "On interpretation of network embedding via taxonomy induction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1812–1820.
[3] A. Dalmia, G. J, and M. Gupta, "Towards interpretation of node embeddings," in *Proc. Web Conf.*, Lyon, France, 2018, pp. 945–952.
[4] A. Gogoglou, C. B. Bruss, and K. E. Hines, "On the interpretability and evaluation of graph representation learning," 2019, *arXiv:1910.03081*.
[5] C. V. Cannistraci, G. Alanis-Lobato, and T. Ravasi, "From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks," *Sci. Rep.*, vol. 3, no. 1, 2013, Art. no. 1613.
[6] X. Yue et al., "Graph embedding on biomedical networks: Methods, applications and evaluations," *Bioinformatics*, vol. 36, pp. 1241–1251, Oct. 2019.
[7] T. N. Dong, J. Schrader, S. Mücke, and M. Khosla, "A message passing framework with multiple data integration for miRNA-disease association prediction," *Sci. Rep.*, vol. 12, 2022, Art. no. 16259.
[8] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9240–9251.
[9] T. Funke, M. Khosla, M. Rathee, and A. Anand, "Zorro: Valid, sparse, and stable explanations in graph neural networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8687–8698, Aug. 2023.
[10] M. Vu and M. T. Thai, "PGM-explainer: Probabilistic graphical model explanations for graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 12225–12235.
[11] C. T. Duong, Q. V. H. Nguyen, and K. Aberer, "Interpretable node embeddings with mincut loss," in *Proc. Learn. Reasoning Graph-Struct. Representations Workshop*, 2019.
[12] M. Idahl, M. Khosla, and A. Anand, "Finding interpretable concept spaces in node embeddings using knowledge bases," in *Proc. Int. Workshops ECML PKDD Mach. Learn. Knowl. Discov. Databases*, Springer, 2020, pp. 229–240.
[13] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1135–1144.
[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4765–4774.

[15] M. Girvan and M. E. Newman, "Community structure in social and biological networks," in *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, 2002.

[16] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Netw.*, vol. 5, no. 2, pp. 109–137, 1983.

[17] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.

[18] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

[19] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, Mar. 2017.

[20] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1105–1114.

[21] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 891–900.

[22] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.

[23] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: A comprehensive review," *Comput. Social Netw.*, vol. 6, no. 1, pp. 1–23, 2019.

[24] S. Bonner, I. Kureshi, J. Brennan, G. Theodoropoulos, A. S. McGough, and B. Obara, "Exploring the semantic content of unsupervised graph embeddings: An empirical study," *Data Sci. Eng.*, vol. 4, pp. 269–289, Sep. 2019.

[25] S. Khoshraftar, S. Mahdavi, and A. An, "Centrality-based interpretability measures for graph embeddings," in *Proc. IEEE 8th Int. Conf. Data Sci. Adv. Analytics*, 2021, pp. 1–10.

[26] Y. Wang, Y. Yao, H. Tong, F. Xu, and J. Lu, "Discerning edge influence for network embedding," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 429–438.

[27] H. Park and J. Neville, "Generating post-hoc explanations for skip-gram-based node embeddings by identifying important nodes with bridgeness," *Neural Netw.*, vol. 164, pp. 546–561, 2023.

[28] C. T. Duong, T. T. Nguyen, T.-D. Hoang, H. Yin, M. Weidlich, and Q. V. H. Nguyen, "Deep MinCut: Learning node embeddings by detecting communities," *Pattern Recognit.*, vol. 134, 2023, Art. no. 109126.

[29] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton, "GEMSEC: Graph embedding with self clustering," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2019, pp. 65–72.

[30] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.

[31] B. Kang, J. Lijffijt, and T. De Bie, "ExplaiNE: An approach for explaining network embedding-based link predictions," 2019, *arXiv: 1904.12694*.

[32] S. Zhang et al., "PaGE-Link: Path-based graph neural network explanation for heterogeneous link prediction," in *Proc. ACM Web Conf.*, 2023, pp. 3784–3793.

[33] Z. Wang, B. Zong, and H. Sun, "Modeling context pair interaction for pairwise tasks on graphs," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 851–859.

[34] A. Rossi, D. Firmani, P. Merialdo, and T. Teofili, "Explaining link prediction systems based on knowledge graph embeddings," in *Proc. Int. Conf. Manage. Data*, 2022, pp. 2062–2075.

[35] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen, "Interaction embeddings for prediction and explanation in knowledge graphs," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 96–104.

[36] T. Prouteau, N. Dugué, N. Camelin, and S. Meignier, "Are embedding spaces interpretable? Results of an intrusion detection evaluation on a large french corpus," in *Proc. 13th Lang. Resour. Eval. Conf.*, 2022, pp. 4414–4419.

[37] L. K. Şenel, I. Utlu, V. Yücesoy, A. Koc, and T. Cukur, "Semantic structure and interpretability of word embeddings," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 10, pp. 1769–1779, Oct. 2018.

[38] J. Shin, A. Madotto, and P. Fung, "Interpreting word embeddings with eigenvector analysis," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018.

[39] S. Park, J. Bak, and A. Oh, "Rotated word vector representations and their interpretability," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Copenhagen, Denmark, 2017, pp. 401–411.

[40] P. P. Liang, M. Zaheer, Y. Wang, and A. Ahmed, "Anchor & transform: Learning sparse embeddings for large vocabularies," 2020, *arXiv:2003.08197*.

[41] F. Sun, J. Guo, Y. Lan, J. Xu, and X. Cheng, "Sparse word embeddings using L1 regularized online learning," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 2915–2921.

[42] H. Luo, Z. Liu, H. Luan, and M. Sun, "Online learning of interpretable word embeddings," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1687–1692.

[43] A. Subramanian, D. Pruthi, H. Jhamtani, T. Berg-Kirkpatrick, and E. Hovy, "SPINE: SParse interpretable neural embeddings," in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 1.

[44] Y. Chen and M. J. Zaki, "KATE: K-competitive autoencoder for text," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 85–94.

[45] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. A. Smith, "Sparse overcomplete word vector representations," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, 2015, pp. 1491–1500.

[46] V. Belle and I. Papantonis, "Principles and practice of explainable machine learning," *Front. Big Data*, vol. 4, 2021, Art. no. 39.

[47] A. Datta, S. Sen, and Y. Zick, "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems," in *Proc. IEEE Symp. Secur. Privacy*, 2016, pp. 598–617.

[48] J. Li, W. Monroe, and D. Jurafsky, "Understanding neural networks through representation erasure," 2016, *arXiv:1612.08220*.

[49] P. Kolpaczki, V. Bengs, M. Muschalik, and E. Hüllermeier, "Approximating the shapley value without marginal contributions," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 13246–13255.

[50] J. Zhang, Q. Sun, J. Liu, L. Xiong, J. Pei, and K. Ren, "Efficient sampling approaches to shapley value approximation," in *Proc. ACM Manage. Data*, vol. 1, no. 1, pp. 1–24, 2023.

[51] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowl. Inf. Syst.*, vol. 41, no. 3, pp. 647–665, 2014.

[52] K. Aas, M. Jullum, and A. Løland, "Explaining individual predictions when features are dependent: More accurate approximations to shapley values," *Artif. Intell.*, vol. 298, 2021, Art. no. 103502.

[53] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5782–5799, May 2023.

[54] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proc. ICML Workshop Unsupervised Transfer Learn.*, 2012, pp. 37–49.

[55] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 587–596.

[56] N. Bansal, X. Chen, and Z. Wang, "Can we gain more from orthogonality regularizations in training deep networks?," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4266–4276.

[57] E. Massart, "Orthogonal regularizers in deep learning: How to handle rectangular matrices?," in *Proc. 26th Int. Conf. Pattern Recognit.*, 2022, pp. 1294–1299.

[58] N. Schaaf, M. Huber, and J. Maucher, "Enhancing decision tree based interpretation of deep neural networks through L1-orthogonal regularization," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl.*, 2019, pp. 42–49.

[59] I. Higgins et al., "Towards a definition of disentangled representations," 2018, *arXiv: 1812.02230*.

[60] I. Higgins et al., "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[61] J. Cha and J. Thiyagalingam, "Orthogonality-enforced latent space in autoencoders: An approach to learning disentangled representations," in *Proc. 40th Int. Conf. Mach. Learn.*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., 2023, pp. 3913–3948.

[62] Y. Song, N. Sebe, and W. Wang, "Orthogonal SVD covariance conditioning and latent disentanglement," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, pp. 8773–8786, Jul. 2023.

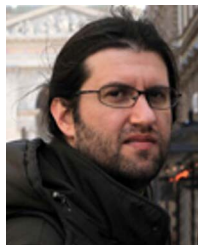
[63] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 874–883.

[64] Y. Liang et al., "Can a fruit fly learn word embeddings?," 2021, *arXiv:2101.06887*.

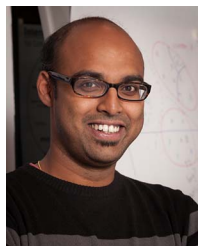
[65] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *J. Mach. Learn. Res.*, vol. 24, no. 127, pp. 1–21, 2023.
[66] R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. S. Bhowmick, "Scaling attributed network embedding to massive graphs," in *Proc. VLDB Endowment*, vol. 14, no. 1, pp. 37–49, 2020.
[67] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Sci. Adv.*, vol. 3, no. 5, 2017, Art. no. e1602548.
[68] D. Hric, R. K. Darst, and S. Fortunato, "Community detection in networks: Structural communities versus ground truth," *Phys. Rev. E*, vol. 90, no. 6, 2014, Art. no. 062805.
[69] Z. Ding, X. Zhang, D. Sun, and B. Luo, "Overlapping community detection based on network decomposition," *Sci. Rep.*, vol. 6, no. 1, pp. 1–11, 2016.
[70] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, no. 3–5, pp. 75–174, 2010.
[71] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 7307, pp. 761–764, 2010.
[72] T. S. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," *Phys. Rev. E*, vol. 80, no. 1, 2009, Art. no. 016105.
[73] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statist. Mechanics Theory Experiment*, vol. 2008, no. 10, 2008, Art. no. P10008.
[74] S. Chanpuriya and C. Musco, "InfiniteWalk: Deep network embeddings as laplacian embeddings with a nonlinearity," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1325–1333.
[75] G. Salha-Galvan, J. F. Lutzeyer, G. Dasoulas, R. Hennequin, and M. Vazirgiannis, "Modularity-aware graph autoencoders for joint community detection and link prediction," *Neural Netw.*, vol. 153, pp. 474–495, 2022.
[76] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, pp. 68–77, 2019.
[77] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip! Online learning of multi-scale network embeddings," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2017, pp. 258–265.
[78] A. Clauset, C. Moore, and M. E. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, pp. 98–101, 2008.
[79] Z. Wu, X. Lin, Z. Lin, Z. Chen, Y. Bai, and S. Wang, "Interpretable graph convolutional network for multi-view semi-supervised learning," *IEEE Trans. Multimedia*, vol. 25, pp. 8593–8606, 2023, doi: 10.1109/TMM.2023.3260649.

Simone Piaggesi received the PhD degree in data science and computation from the University of Bologna, Italy, in 2023. He is currently a post-doctoral fellow with the Computer Science Department, University of Pisa, Italy. His research interests include interpretability of graph learning models.

Megha Khosla is an assistant professor with the Intelligent Systems Department, TU Delft, The Netherlands. Her main research area is machine learning on graphs with focus on three key aspects of effectiveness, interpretability, and privacy-preserving learning.



André Panisson is a principal researcher with the CENTAI Institute in Turin, Italy. His current research focuses on the development of tools to facilitate the explainability, fairness, and transparency of artificial intelligence systems. His past and current research also focuses on the intersection of machine learning, network science, and data science, primarily on developing methods for the analysis, modeling, and simulation of complex phenomena in systems that involve technological and social factors.



Avishek Anand is an associate professor with the Web Information Systems (WIS), Software Technology (ST) Department, Delft University of Technology (TU Delft). He is also a member of the L3S Research Center, Hannover, Germany. One of his main research focus is interpretability of machine learning models with focus on representations from discrete input like text and graphs.

Open Access provided by 'Università di Pisa' within the CRUI CARE Agreement