Modularity-aware graph autoencoders for joint community detection and link prediction

Guillaume Salha-Galvan, Johannes Lutzeyer, George Dasoulas, Romain Hennequin, Michalis Vazirgiannis

Modularity-Aware Graph Autoencoders for Joint Community Detection and Link Prediction

Guillaume Salha-Galvan[a,b,*], Johannes F. Lutzeyer[b], George Dasoulas[b], Romain Hennequin[a], Michalis Vazirgiannis[b,c]

[a] *Deezer Research, Paris, France*
[b] *LIX, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France*
[c] *Athens University of Economics and Business (AUEB), Athens, Greece*

Abstract

Graph autoencoders (GAE) and variational graph autoencoders (VGAE) emerged as powerful methods for link prediction. Their performances are less impressive on community detection problems where, according to recent and concurring experimental evaluations, they are often outperformed by simpler alternatives such as the Louvain method. It is currently still unclear to which extent one can improve community detection with GAE and VGAE, especially in the absence of node features. It is moreover uncertain whether one could do so while simultaneously preserving good performances on link prediction. In this paper, we show that jointly addressing these two tasks with high accuracy is possible. For this purpose, we introduce and theoretically study a community-preserving message passing scheme, doping our GAE and VGAE encoders by considering both the initial graph structure and modularity-based prior communities when computing embedding spaces. We also propose novel training and optimization strategies, including the introduction of a modularity-inspired regularizer complementing the existing reconstruction losses for joint link prediction and community detection. We demonstrate the empirical effectiveness of our approach, referred to as Modularity-Aware GAE and VGAE, through in-depth experimental validation on various real-world graphs.

Keywords: Graph Autoencoders, Node Embedding, Modularity, Graph Neural Networks, Link Prediction, Community Detection

1. Introduction

Context and Motivation. Graph structures became ubiquitous in various fields ranging from social networks and web mining to biology [1, 2], due to the proliferation of data representing entities, also known as (a.k.a.) *nodes*, connected by links, a.k.a. *edges*. Extracting relevant information from these nodes and edges is essential to tackle a wide range of graph-based machine learning problems [1, 2, 3, 4], such as the challenging tasks of:

*Corresponding author: research@deezer.com

- *link prediction* [5, 6], which consists in inferring the presence of new or unobserved edges between some pairs of nodes, based on observable edges in the graph;
- *community detection* [7, 8], which consists in clustering nodes into similar subgroups, according to a chosen similarity metric.

To effectively address such graph-based problems, significant research efforts were recently devoted to the development of *node embedding* methods. In a nutshell, these methods aim to learn vectorial representations of nodes, in an *embedding space* where node positions should reflect and summarize the initial graph structure [1, 2, 9]. Then, they assess the probability of a missing edge between two nodes, or their likelihood of belonging to the same community, by evaluating the proximity of these nodes in the learned space [10, 11, 12]. Node embedding methods are at the core of promising improvements in real-world graph learning applications [2, 6], often outperforming more traditional graph mining methods relying on hand-engineered indicators [5, 6].

In particular, *graph autoencoders* (GAE) and *variational graph autoencoders* (VGAE) [10, 11, 13, 14] recently emerged as two powerful families of node embedding methods. They both rely on an *encoding-decoding* strategy that, in a broad sense, consists in *encoding* nodes into an embedding space from which *decoding*, i.e., reconstructing the original graph should ideally be possible, by leveraging either a deterministic (for GAE) or a probabilistic (for VGAE) approach. Originally mainly designed for link prediction (at least in their modern formulation leveraging *graph neural networks* architectures [11]), the overall effectiveness of GAE and VGAE and of their extensions on this specific task has been widely experimentally confirmed over the past few years [15, 16, 17, 18, 19, 20, 21, 22].

On the other hand, several concurring studies [12, 17, 23, 24] have simultaneously pointed out the limitations of these models on community detection. They emphasized that standard GAE and VGAE are often outperformed by simpler node clustering alternatives, such as the popular Louvain method [7]. While some recent studies worked on this issue (see Section 2 for an overview), their solutions strongly relied on *clustering-oriented probabilistic* priors that only fit the VGAE setting and can not be directly transposed to GAE. They also benefited greatly from the presence of *node features* a.k.a. *attributes* complementing the graph structure, but provided only little to no empirical gain on featureless graphs that are nonetheless ubiquitous. Thirdly, they did not explicitly try to preserve the good performances of GAE and VGAE on link prediction. In practice, as we will argue in this work, learning node embedding spaces that jointly enable good link prediction and community detection performances is often desirable, both for real-world applications and in pursuit of learning accurate and general representations of a graph structure.

Research Questions. In summary, the question of how to improve community detection with GAE and VGAE remains incompletely addressed, especially in the absence of node features, and it is still unclear to which extent one can improve community detection with these models without simultaneously deteriorating link prediction. In this paper, we propose to tackle these important problems by investigating the following two research questions:

- **Question 1:** Can we improve community detection for *both* the GAE *and* VGAE settings? And does this improvement persist for *featureless* graphs?

- **Question 2:** Do improvements in the community detection task necessarily incur a loss in the link prediction performance or can they be *jointly* addressed with high accuracy?

Contributions. In this paper, we propose several novel contributions to both the GAE and VGAE frameworks, which allow us to answer both of these research questions positively. More precisely, our contributions are listed as follows.

1. We first diagnose the reasons why GAE and VGAE models tend to perform well on link prediction but to underperform on community detection.
2. Then, based on insights from this diagnosis, we improve GAE and VGAE for community detection while preserving their ability to identify missing edges. Our strategy leverages concepts inspired by *modularity-based* clustering [7, 25, 26]:
   (a) Specifically, we first present and theoretically study a novel *community-preserving message passing scheme*, doping our GAE and VGAE encoders by considering both the initial graph structure and modularity-based prior communities when computing embedding spaces;
   (b) We also introduce revised training and optimization strategies with respect to (w.r.t.) current practices in the scientific literature, including the introduction of modularity-inspired losses complementing the existing reconstruction losses with the aim of jointly ensuring good performances on link prediction and community detection.
3. Backed by in-depth experiments on several real-world graphs, including on industrial-scale data provided by a global music streaming service, we demonstrate the empirical effectiveness of our approach at addressing:
   (a) Pure community detection problems;
   (b) Joint community detection and link prediction problems.
4. Lastly, along with this paper, we publicly release our source code on GitHub, to ensure the reproducibility of our results and to encourage future usage of our method.

Organization of this Paper. The remainder of this paper is organized as follows. In Section 2, we recall key concepts related to GAE and VGAE models, as well as their existing applications to link prediction and to community detection. We also point out the limits of current GAE and VGAE models on community detection. In Section 3, we diagnose the reasons explaining these limits. We subsequently introduce and theoretically study our proposed solution, referred to as *Modularity-Aware GAE and VGAE*, to overcome these limitations. We report and discuss our experimental evaluation in Section 4, and we conclude in Section 5.

2. Background and Related Work

We begin this section by providing a description of the GAE and VGAE models in Sections 2.1 and 2.2, respectively. We then introduce the two learning tasks we address in this work and previous solution approaches. Specifically, we discuss the link prediction task in Section 2.3 and the community detection task in Section 2.4.

Throughout this paper, we consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges. We denote by A the $n \times n$ adjacency matrix of $\mathcal{G}$, that is either:

Table 1: Overview of frequently used notation.

Notation	Domain	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	–	A graph composed of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$
n, m	$\mathbb{N}^+$	Number of nodes and edges in $\mathcal{G}$, respectively
f	$\mathbb{N}^+$	Dimension of feature vectors describing nodes
x_i	$\mathbb{R}^f$	Feature vector describing node $i \in \mathcal{V}$
$X = [x_1, \ldots, x_n]^T$	$\mathbb{R}^{n \times f}$	Node feature matrix, stacking up x_i vectors of all nodes
A	$[0,1]^{n \times n}$	Adjacency matrix of $\mathcal{G}$
$D = \mathrm{diag}(A\mathbf{1}_n)$	$\mathbb{R}^{n \times n}$	Degree matrix corresponding to the adjacency matrix A
$\mathcal{F}_{GCN}(A)$	$[0,1]^{n \times n}$	Symmetric normalization of A defined in Equation (3), and used as message passing operator in Graph Convolutional Networks (GCN)
d	$\mathbb{N}^+$	Dimension of the node embedding space learned by GAE and VGAE models
z_i	$\mathbb{R}^d$	Node embedding vector of node $i \in \mathcal{V}$
$Z = [z_1, \ldots, z_n]^T$	$\mathbb{R}^{n \times d}$	Node embedding matrix, stacking up z_i vectors of all nodes
$\hat{A}$	$[0,1]^{n \times n}$	Adjacency matrix reconstructed by the GAE or VGAE
$\mathcal{L}_{\mathrm{GAE}}$	$\mathbb{R}^+$	Reconstruction loss minimized in standard GAE
$\mathcal{L}_{\mathrm{VGAE}}$	$\mathbb{R}^-$	Evidence lower bound maximized in standard VGAE
$\tilde{\mathcal{L}}_{\mathrm{GAE}}$	$\mathbb{R}$	Revised loss minimized in our Modularity-Aware GAE
$\tilde{\mathcal{L}}_{\mathrm{VGAE}}$	$\mathbb{R}$	Revised objective maximized in our Modularity-Aware VGAE
$C_1, \ldots, C_K$	$\mathcal{V}$	A partition of $\mathcal{V}$ into K disjoint node clusters/communities
$n_1, \ldots, n_K$	$\{1, ..., n\}$	Number of nodes in $C_1, \ldots, C_K$ respectively
A_c	$\{0,1\}^{n \times n}$	Community membership matrix defined in Equation (14), corresponding to the adjacency matrix of a graph defined by the $C_1, \ldots, C_K$ partition
A_s	$\{0,1\}^{n \times n}$	s-regular sparsification of A_c, in which each node in C_l is only connected to $s < n_l$ randomly selected nodes in C_l
λ	$\mathbb{R}^+$	Hyperparameter regulating the relative importance of A_s in encoders of the Modularity-Aware GAE and VGAE
β, γ	$\mathbb{R}^+$	Hyperparameters regulating the modularity component in the revised losses of the Modularity-Aware GAE and VGAE

- binary, i.e., for all $(i, j) \in \mathcal{V} \times \mathcal{V}$, we have $A_{ij} \in \{0, 1\}$;
- or weighted and normalized, i.e., for all $(i, j) \in \mathcal{V} \times \mathcal{V}$, we have $A_{ij} \in [0, 1]$.

Each node $i \in \mathcal{V}$ is also equipped with an f-dimensional feature vector x_i. In the following, X denotes the $n \times f$ feature matrix stacking up all feature vectors. When dealing with featureless graphs, we will simply set $X = I_n$, where I_n denotes the $n \times n$ identity matrix. In Table 1 we provide an overview of our frequently used notation.

2.1. ***Graph Autoencoders***

The term *graph autoencoders* (GAE) refers to a family of unsupervised models learning node embedding spaces from graph data [9, 11, 13, 14, 27]. They involve the combination of two components, an *encoder* and a *decoder*, that jointly learn low-dimensional vectorial

representations of nodes from which one should be able to reconstruct the initial graph. Intuitively, the ability to accurately reconstruct a graph from a node embedding space indicates that this space preserves some important information about the graph structure.

Albeit under various formulations, this encoding-decoding strategy has been widely adopted over the last years to learn node embedding spaces in the absence of node labels [9, 10, 11, 13, 14]. In this section, we choose to mainly follow the formulation of Kipf and Welling [11], as their work is explicitly mentioned as the seminal reference in the majority of the most recent advances in GAE, including [12, 15, 17, 18, 19, 20, 21, 22, 23, 27, 28, 29].

2.1.1. Encoder

The first component of a GAE model is an *encoder*. In its most general formulation, it is a parameterized function processing A and X, and aiming to map each node $i \in \mathcal{V}$ from $\mathcal{G}$ to a low-dimensional *embedding vector* $z_i \in \mathbb{R}^d$, with $d \ll n$. Denoting by Z the $n \times d$ matrix stacking up all vectors z_i, we have:

$$Z = \text{Encoder}(A, X). \tag{1}$$

In practice, a *graph neural network* architecture [2, 3] often acts as the encoder. In particular, Kipf and Welling [11] leverage multi-layer *graph convolutional network* (GCN) encoders [30]. In a L-layer GCN, with $L \geq 2$, with input layer $H^{(0)} = X$, and with output layer $H^{(L)} = Z$, we have:

$$\begin{cases} H^{(0)} = X, \\ H^{(l)} = \text{ReLU}(\mathcal{F}_{GCN}(A)H^{(l-1)}W^{(l-1)}), \quad \text{for } l \in \{1, ..., L-1\}, \\ H^{(L)} = Z = \mathcal{F}_{GCN}(A)H^{(L-1)}W^{(L-1)}, \end{cases} \tag{2}$$

where

$$\mathcal{F}_{GCN}(A) = (D + I_n)^{-\frac{1}{2}}(A + I_n)(D + I_n)^{-\frac{1}{2}} \tag{3}$$

is the *symmetric normalization* of the adjacency matrix A, and where $D = \text{diag}(A\mathbf{1}_n)$ ($\mathbf{1}_n$ denotes the vector containing n entries all equal to 1) is the diagonal degree matrix of A. At each layer $l > 1$, the GCN computes a vectorial representation for each node $i \in \mathcal{V}$, by computing a weighted average of the representations from layer $l-1$ of i's direct neighbors and of i itself. This averaging operation is composed with a linear transformation via trainable weight matrices $W^{(0)}, \ldots, W^{(L-1)}$ and a ReLU activation function: $\text{ReLU}(x) = \max(x, 0)$. The tuning of these *weight matrices*, as proposed in [11], will be detailed in Section 2.1.3.

To this day, GCNs remain popular encoders in GAE extensions [12, 15, 18, 19, 20, 21, 22, 28, 24] building upon Kipf and Welling [11]. This can be explained by the recent successes of GCN-based models [2, 3, 30], as well as by the simplicity of GCNs in comparison to other graph neural networks [3, 31] and their linear time complexity w.r.t. the number of edges m in the graph [30]. Nonetheless, the choice of GCN encoders is made without loss of generality, as they can be replaced by alternatives, including by faster [32, 33, 34], by more sophisticated [31, 35, 36] or, on the contrary, by simpler [23, 27] models.

2.1.2. Decoder

The second component of a GAE is a *decoder*. This function aims to reconstruct an $n \times n$ adjacency matrix $\hat{A}$, estimated from the embedding vectors:

$$\hat{A} = \text{Decoder}(Z). \tag{4}$$

While another neural network could act as a decoder [14, 37, 38], Kipf and Welling [11] and most of the aforementioned extensions rely on simpler *inner product decoders*:

$$\hat{A} = \sigma(ZZ^T), \tag{5}$$

where $\sigma(\cdot)$ denotes the sigmoid function $\sigma(x) = 1/(1+e^{-x})$. Therefore, for all node pairs $(i,j) \in \mathcal{V} \times \mathcal{V}$, we have $\hat{A}_{ij} = \sigma(z_i^T z_j) \in [0,1]$. In such a setting, a large and positive inner product $z_i^T z_j$ in the node embedding space indicates the likely presence of an edge between nodes i and j in $\mathcal{G}$, according to the model. Again, the choice of inner product decoders is made without loss of generality, and recent efforts considered replacing them by alternatives verifying some desirable properties, such as the ability to reconstruct directed edges [18], to capture triads structures [22] or to reconstruct biologically plausible graphs in the case of autoencoders for molecular structures [39, 40].

2.1.3. Optimization

Recall that GAE models aim to learn node embedding spaces from which one can accurately reconstruct graphs. They are trained to minimize *reconstruction losses*, that evaluate the similarity between the decoded adjacency matrix $\hat{A}$ and the original one A. Specifically, Kipf and Welling [11] train weight matrices of their GCN encoders by iteratively minimizing, using gradient descent [41], the following *cross-entropy* loss:

$$\mathcal{L}_{\text{GAE}} = \frac{-1}{n^2} \sum_{(i,j) \in \mathcal{V} \times \mathcal{V}} \Big[A_{ij} \log(\hat{A}_{ij}) + (1 - A_{ij}) \log(1 - \hat{A}_{ij}) \Big]. \tag{6}$$

In the case of sparse graphs where unconnected node pairs significantly outnumber the connected ones, i.e., the graph's edges, it is common to reweight the "positive terms" in Equation (6) by a factor $w_{\text{pos}} > 1$ [11, 24], or alternatively to subsample "negative terms" [9, 42]. We also note that an exact evaluation of $\mathcal{L}_{\text{GAE}}$ requires the reconstruction of the entire matrix $\hat{A}$, which suffers from a quadratic $O(dn^2)$ time complexity. For scalability concerns, recent works proposed faster training strategies [17, 24, 42]. This includes the FastGAE method [24] that approximates $\mathcal{L}_{\text{GAE}}$ by reconstructing stochastic *subgraphs* of $O(n)$ size, and that we will also use in Section 4 in our experiments on large graphs.

2.2. ***Variational Graph Autoencoders***

Kipf and Welling [11] also considered a probabilistic variant of GAEs, extending *variational autoencoders* (VAE) from Kingma and Welling [43]. Besides constituting generative models with promising recent applications to graph generation [39, 40, 44], variants of *variational graph autoencoders* (VGAE) also turned out to be effective alternatives to GAE in some link prediction or community detection tasks [11, 18, 21, 23, 24, 27]. Consequently, we see value in considering both GAE and VGAE in our work.

2.2.1. Encoder

VGAE models provide an alternative strategy to learn a matrix Z, stacking up one embedding vector z_i for each node i of a graph $\mathcal{G}$, by assuming that these vectors are drawn from specific distributions. In particular, Kipf and Welling [11] assume that each vector z_i is a sample drawn from a d-dimensional Gaussian distribution, with mean vector $\mu_i \in \mathbb{R}^d$ and variance matrix $\text{diag}(\sigma_i^2) \in \mathbb{R}^{d \times d}$ (with $\sigma_i \in \mathbb{R}^d$). They rely on *two encoders* to learn these parameters. Denoting the $n \times d$ matrices stacking up the d-dimensional mean and (log)-variance vectors for each node by μ and by $\log \sigma$, respectively, they set:

$$\mu = \text{Encoder}_\mu(A, X) \text{ and } \log \sigma = \text{Encoder}_\sigma(A, X). \tag{7}$$

As is the case for GAE, multi-layer GCNs often act as encoders, i.e., $\mu = \text{GCN}_\mu(A, X)$ and $\log \sigma = \text{GCN}_\sigma(A, X)$. Then, they adopt a *mean-field inference model* for Z [11], i.e.,

$$q(Z \mid A, X) = \prod_{i=1}^{n} q(z_i \mid A, X), \text{ with } q(z_i \mid A, X) = \mathcal{N}(z_i \mid \mu_i, \text{diag}(\sigma_i^2)), \tag{8}$$

where $\mathcal{N}(\cdot)$ denotes the normal distribution.

2.2.2. Decoder

In the VGAE setting, the actual embedding vectors z_i are sampled from the aforementioned normal distributions. From such embedding representations, VGAE models then require a *generative model* $p(A \mid Z, X)$, to act as a graph *decoder*. As for GAE, Kipf and Welling [11] rely on inner products together with sigmoid activation functions to reconstruct edges:

$$\hat{A}_{ij} = p(A_{ij} = 1 \mid z_i, z_j) = \sigma(z_i^T z_j), \tag{9}$$

where the embeddings z_i, z_j are sampled from the distribution in Equation (8). Then, the authors assume the following generative model which factorizes over the edges:

$$p(A \mid Z, X) = \prod_{i=1}^{n} \prod_{j=1}^{n} p(A_{ij} \mid z_i, z_j). \tag{10}$$

2.2.3. Optimization

During training, and similarly to standard VAE models [43], Kipf and Welling [11] iteratively maximize a tractable variational lower bound of the model's likelihood, a.k.a. the *evidence lower bound* (ELBO) [43], written as follows in the context of VGAE:

$$\mathcal{L}_{\text{VGAE}} = \mathbb{E}_{q(Z|A,X)}\Big[\log p(A \mid Z, X)\Big] - \mathcal{D}_{KL}\Big(q(Z \mid A, X)||p(Z)\Big). \tag{11}$$

This ELBO is iteratively maximized w.r.t. weights of the two GCN encoders, by gradient descent and potentially using approximation strategies such as the strategies described in Section 2.1.3. In the above Equation (11), $\mathcal{D}_{KL}(\cdot||\cdot)$ denotes the Kullback-Leibler divergence [45], and $p(Z)$ corresponds to a unit Gaussian prior on the distribution of the latent vectors, that can also be interpreted as a regularization term on the magnitude of the embedding vectors. We refer to [9, 43, 46] for complete details and derivations of such ELBO bounds in the context of VAE models.

2.3. ***Evaluating GAE and VGAE: Link Prediction***

The question of how to properly determine the quality of node embedding representations learned from GAE and VGAE models is crucial. While one could directly report reconstruction losses [14], recent research work instead strives to apply the GAE and VGAE models to *downstream evaluation tasks*, which permit reporting more insightful and interpretable evaluation metrics [14, 13, 11]. In particular, Kipf and Welling [11] evaluate their GAE and VGAE models on *link prediction* problems in citation networks [47].

2.3.1. The Link Prediction Task

Kipf and Welling [11] follow an evaluation methodology consisting of:

- Training their models on an *incomplete* version of an original graph, for which only a certain percentage of randomly sampled edges (85% in their case) are visible;
- Constructing *validation* and *test sets* gathering:
  - node pairs corresponding to missing edges (5% and 10%, respectively, in [11]);
  - the same number of randomly picked unconnected node pairs in the graph;
- Evaluating the models' abilities to distinguish edges from non-edges in these sets, using node embedding representations learned on the incomplete graph.

Indeed, while all node pairs in the validation and test sets are observed to be unconnected during training, half of them actually correspond to missing edges from the original graph. Link prediction acts as a binary classification downstream task, evaluating to which extent the decoder's predictions $\hat{A}_{ij} = \sigma(z_i^T z_j)$ correctly locate and reconstruct these missing edges despite their absence during training. Performance in the link prediction task is evaluated using metrics such as the *Area Under the ROC Curve* (AUC) and *Average Precision* (AP) scores [48].

2.3.2. Link Prediction with GAE, VGAE and Extensions

Kipf and Welling [11] show that their proposed GAE and VGAE reach competitive link prediction scores w.r.t. some popular node embedding methods, such as DeepWalk [49] and Laplacian eigenmaps [50]. They also emphasize an additional benefit of the GCN-based GAE and VGAE over baseline methods such as DeepWalk and Laplacian eigenmaps, which is the ability to leverage both the graph structure and node features when learning embedding spaces.

Over the last few years, the overall effectiveness of the GAE and VGAE paradigms at addressing link prediction has been widely confirmed experimentally [15, 16, 17, 18, 19, 20, 21, 22, 24, 27, 28, 51, 52, 53]. Numerous research efforts proposed and evaluated variants of GAE and VGAE designed for this specific task, improving their performances by considering more refined encoders [17, 21, 51, 54], decoders [18, 20, 21, 22] or regularization techniques [15, 19, 28]. Other works also successfully addressed different downstream tasks that are closely related to link prediction, such as edge classification [52] or graph-based recommendation [51, 53, 55].

2.4. *Evaluating GAE and VGAE: Community Detection*

While link prediction remains a prominent evaluation task for GAE and VGAE, they have also shown promising results on (semi-supervised) node classification [16, 21], canonical correlation analysis [56] and, in the case of VGAE, graph generation especially in the context of molecular graph data [39, 40, 57]. However, their performances are less impressive on *community detection* [12, 23], on which we focus in this section.

2.4.1. The Community Detection Task

Among the fundamental problems in graph-based machine learning, *community detection* (which we regard as a synonym of *node clustering* in this work, consistently with Fortunato [58]) consists in identifying $K < n$ clusters a.k.a. *communities* of nodes that, in some sense, are more similar to each other than to the other nodes [8, 12, 58]. More formally, we aim to obtain a partition of the node set $\mathcal{V}$ into K sets:

$$C_1 \subseteq \mathcal{V}, \dots, C_K \subseteq \mathcal{V}, \tag{12}$$

with cardinality $|C_k| = n_k \leq n$ for $k \in \{1, \dots, K\}$. The quality of such a partition is usually assessed through some predefined similarity metrics, e.g., unsupervised density-based metrics[1] calculated from the intra- and inter-cluster edge density [8], or scores such as the normalized *Mutual Information* (MI) [12] that compares the partition to some ground-truth node labels hidden during training.

Improving community detection on graphs has been the objective of significant efforts over the last decades (see, e.g., [8, 58] for a review), and still constitutes an active area of research [12, 59, 60, 61, 62] with numerous applications. This includes the segmentation of websites in a web graph according to thematic categories, as well as the detection of densely connected subgroups of users in online social networks [8].

2.4.2. Community Detection with GAE and VGAE

In the presence of node embedding representations, community detection boils down to the more standard problem of clustering n vectors in a d-dimensional Euclidean space into K groups [63]. With this goal in mind, several studies specifically tried to perform community detection with GAE and VGAE by:

- learning an embedding vector z_i for each $i \in \mathcal{V}$, as described in Sections 2.1 and 2.2;
- clustering the resulting vectors z_i into K groups, through one of the numerous clustering methods for Euclidean data, such as the popular k-means algorithm [63].

However, concurring experimental evaluations [12, 17, 23, 24] recently pointed out the limitations of such an approach. They emphasized its lower performance w.r.t. simpler community detection alternatives, that sometimes even directly operate on the graph structure without considering node features, such as the popular Louvain method [7].

[1] Such metrics usually rely on *homophily* assumptions. This term describes the tendency of nodes to connect to "similar" nodes in the graph, which is observed in numerous real-world applications [6]. Under such assumptions, intuitively, nodes from the same community should be more densely connected, and nodes from different communities should be more sparsely connected.

For instance, Choong et al. [23] show that, on the (featureless) Cora citation network [47], a VGAE+k-means strategy reaches a mean normalized MI score of 23.84%, way below the Louvain method (43.36%). Salha et al. [24] show that, on the same graph, a GAE+k-means also reaches an underwhelming 30.88% mean normalized MI score. These authors obtain comparable conclusions on several other popular graph datasets, such as the featureless versions [47] of Citeseer (9.85% MI for VGAE+k-means vs 16.39% for Louvain, in [24]) and Pubmed (20.41% MI for VGAE+k-means in [23], which is comparable to Louvain, but significantly below the 29.46% MI score obtained by running a k-means on node embedding vectors learned via DeepWalk [49]).

2.4.3. Community Detection with Extensions of GAE and VGAE

Several studies have worked on the issue of the underwhelming performance of GAE and VGAE in the community detection task [12, 23, 29]. Choong et al. [12] introduced VGAECD, a *VGAE for Community Detection (CD)* model that replaces Gaussian priors by learnable *Gaussian mixtures*. Such a choice permits recovering communities from node embedding spaces without relying on an additional k-means step. In a subsequent study [23], the same authors proposed VGAECD-OPT, an improved version of VGAECD. Specifically, VGAECD-OPT replaces GCN encoders with simpler linear models [64], as proposed in Salha et al. [27]. It also adopts a different optimization procedure based on neural expectation-maximization [65], which guarantees that communities do not collapse during training [23] and experimentally leads to better performances.

More recently, Li et al. [29] introduced *Dirichlet Graph Variational Autoencoder* (DGVAE), another extension of VGAE which uses Dirichlet distributions as priors on latent vectors, acting as indicators of community memberships. The *Marginalized GAE* (MGAE) model from Wang et al. [10] is also evaluated on community detection. However, the MGAE model does not explicitly leverage embedding representations for this task; instead the *spectral clustering* [50] is applied to the decoded graphs. Lastly, while community detection was not the main focus in [15, 19, 22, 37, 28], these works all proposed various encoding-decoding methods that, to different extents, seem to outperform standard GAE and VGAE on the community detection task, in the reported evaluations. They consider alternatives encoder or training choices, which we further discuss and investigate in Section 3.

2.4.4. Limitations

While the models discussed in Section 2.4.3 will constitute relevant baselines in our experiments (see Section 4), they still suffer from several fundamental limitations that motivate our work.

- Firstly, all extensions explicitly designed for community detection [12, 23, 29] *rely on clustering-oriented probabilistic priors*. They are only applicable in the VGAE paradigm, and cannot be directly transposed to the deterministic GAE setting. The question of how to design clustering-efficient GAE models thus remains widely open.

- More importantly, a closer look at these models reveals that their *empirical gains often mostly stem from the addition of node features* to the graph. As an illustration, Table 2 displays the reported performances of VGAECD and VGAECD-OPT on the *featureless* versions of two graphs [23]. We observe that they offer little to no

Table 2: Normalized mutual information scores (in %) for community detection on the Cora and Pubmed citation networks, *with* and *without* node features. Results are directly taken from the evaluation of Choong et al. [23]. This table emphasizes that, in the absence of node features, VGAECD and VGAECD-OPT bring little (to no) advantage w.r.t. standard VGAE, and remain below the Deepwalk and/or Louvain baselines. Scores of VGAECD and VGAECD-OPT significantly increase when adding features to the graph. Recall: in this table, Deepwalk and Louvain both ignore node features.

Dataset	**VGAE**	**VGAECD**	**VGAECD-OPT**	**DeepWalk**	**Louvain**
Cora *without* node features	23.84	28.22	37.35	37.96	**43.36**
Pubmed *without* node features	20.41	16.42	25.05	**29.46**	19.83
Cora *with* node features	31.73	50.72	**54.37**	37.96	43.36
Pubmed *with* node features	19.81	32.53	**35.52**	29.46	19.83

empirical advantage when features are absent. This draws into question the extent to which these models are able to capture communities from (only) graph data.

The important role of node features has subsequently been confirmed (e.g., Park et al. [37] show that, on the Pubmed dataset, a straightforward k-means on the node features alone reaches comparable MI scores w.r.t. VGAE and MGAE). On the other hand, most of the aforementioned other studies with empirical improvements [15, 19, 22, 37, 28] only reported results on graphs equipped with node features. This motivates the need for a proper investigation of the *featureless* case where models cannot rely on the additional node feature information.

- Lastly, previous studies centered around community detection [10, 12, 23, 29] *did not explicitly try to preserve the good performances of GAE and VGAE on link prediction.* Overall, most of the aforementioned existing works learn node representations specific to a particular learning task. Therefore, it is still unclear whether one can improve community detection with GAE or VGAE without simultaneously deteriorating link prediction.

  With the general aim of learning high-quality node embeddings, one can wonder to which extent these models can learn representations that are *jointly* useful for several tasks. Besides providing a more accurate summary of the graph structure under consideration, such representations could also lead to significant resource savings in real-world applications. As an illustration, our experiments in Section 4 will consider industrial-scale graph data obtained from the music streaming service Deezer, providing a concrete example of an application for which learning representations jointly effective at link prediction and community detection is desirable.

In conclusion to this section, the question of how to effectively improve community detection with GAE and VGAE remains incompletely addressed.

3. Modularity-Aware (Variational) Graph Autoencoders

We now introduce our approach, referred to as *Modularity-Aware GAE and VGAE* in the following, to address the aforementioned limitations. In Section 3.1, we first provide a general overview of the key components of our solution. They transpose concepts from *modularity-based* clustering [7, 25, 26] to GAEs and VGAEs, and are illustrated in Figure 1. We subsequently detail these solution components in Sections 3.2 and 3.3.

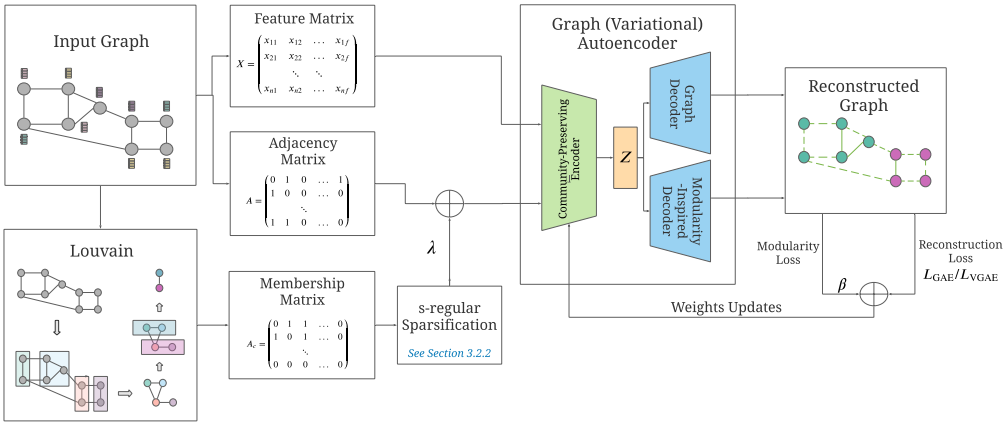


Figure 1: Overview of our proposed *Modularity-Aware GAE/VGAE* model. Firstly, input graph data A and X are combined with the s-regular sparsified prior community membership matrix A_s, derived through iterative modularity maximization via the Louvain algorithm, as described in Section 3.2. Then, they are processed by our revised community-preserving (Linear or GCN) encoders, encoding each node i as an embedding vector z_i of dimension $d \ll n$. Neural weights of encoders are optimized through a procedure combining reconstruction and modularity-inspired losses, and described in Section 3.3.1. Furthermore, other hyperparameters from this model are tuned via the method described in Section 3.3.2 and designed for joint link prediction and community detection applications.

3.1. ***Diagnosis and Overview of our Proposed Solution***

Based on our literature review, we diagnose three main reasons that can explain why previous GAE and VGAE models still suffer from the limitations described in Section 2.4.4.

- Firstly, they leveraged *encoders* that were not specifically designed to preserve the intrinsic communities from the graph structure under consideration in the node embedding space. This includes the popular GCN, as well as refined neural models that rather aimed to preserve clusters from node features (but not necessarily the actual communities from the graph under consideration).

  In *Modularity-Aware GAE and VGAE*, we overcome this issue by incorporating a novel encoding scheme for graph community-preserving representation learning. It consists in an improvement of the GCN *message passing operator*, boosting both GAE and VGAE models by simultaneously considering the initial graph structure and *modularity-based node communities* when computing node embedding spaces. We present and theoretically study this encoder in Section 3.2.

- Besides the encoder's architecture, previous models were often *optimized* in a fashion that, by design, favors link prediction over community detection. In particular, the standard cross-entropy (Equation (6)) and ELBO (Equation (11)) losses, used to learn neural weight matrices, directly involve the reconstruction of *node pairs* from the embedding space[2]. However, as we will detail, a good reconstruction of

[2]In the case of the probabilistic VGAE paradigm, another limitation of the ELBO loss - and of the

local pairwise connections does not necessarily imply a good reconstruction of the *global* community structure.

In *Modularity-Aware GAE and VGAE*, we instead optimize an alternative loss inspired by the *modularity* [7]. Such a loss acts as a simple yet effective global regularization over pairwise reconstruction losses, with desirable properties for joint link prediction and community detection. It will empirically enable a refined optimization of the weight matrices from our encoders. We present this aspect in more details in Section 3.3.1.

- Lastly, in addition to these weight matrices, GAE and VGAE models involve several other hyperparameters, ranging from the number of training iterations to the learning rate [11]. While they also impact the model performance, the selection procedure for such hyperparameters was sometimes omitted in previous works [23] or based on link prediction validation sets [17, 24] (while, intuitively, the best hyperparameters for community detection might differ from those for link prediction).

  For the *Modularity-Aware GAE and VGAE* we adopt an alternative graph-based model selection procedure. It completes the previous two aspects, by providing the most relevant GAE/VGAE hyperparameters for joint link prediction and community selection. We present and discuss this procedure in Section 3.3.2.

3.2. ***Community-Preserving Encoders for GAE and VGAE***

Following this diagnosis and overview, we now provide more detail on the first of the three bullet points in Section 3.1, i.e., our proposed revised *encoding* strategy. We recall that our proposed solution aims to encode nodes as embedding vectors z_i *more suitable for community detection*. Essentially, intrinsic communities in the graph under consideration should be easily retrievable from these representations, e.g., from their L_2 distances via a straightforward k-means clustering. These vectors should also simultaneously remain relevant for *link prediction,* i.e., as for existing GAE and VGAE, the likelihood of a missing edge between two nodes should also be inferred from the learned representations z_i. In the following, for consistency with previous work (see Section 2), we continue using the inner product $\hat{A}_{ij} = \sigma(z_i^T z_j) \in [0, 1]$ as the probability of an edge between nodes i and j.

3.2.1. Revising the Message Passing Operator

Existing graph encoders usually involve normalized versions of the adjacency matrix A, or some generalized *message passing operator* matrix that also captures each node's direct connections in the graph under consideration [66]. For instance, in the popular multi-layer GCN in Equation (2), the symmetric normalization $\mathcal{F}_{GCN}(A)$ from Equation (3) is used such that at each layer l a vectorial representation for each node is computed by taking

underlying generative decoder - lies in the use of standard Gaussian priors. Replacing these priors by for example *Gaussian mixtures* as in [12, 23], appears to be an intuitive approach for community-based learning. However, as this approach 1) does not extend to deterministic GAE, and 2) has been extensively studied in [12, 23], we do not further develop it in this work. We will nonetheless compare to [12, 23] in experiments, and will argue in Section 3.3.1 that Gaussian mixtures could straightforwardly be incorporated in our proposed Modularity-Aware VGAE.

a weighted average of the representations from layer $l-1$ of its direct neighbors and of itself. In this work, we adopt an alternative strategy that consists in computing the weighted average of, at each layer:

- representations from the direct neighbors of each node, as above;
- but also representations from other *unconnected nodes* that, according to some prior available knowledge and criteria, belong to the same graph community.

More precisely, let us assume that we have, at our disposal, a preprocessing *graph mining* technique that, based on the graph structure and on some fixed criteria, learns an initial *prior partition of the node set* $\mathcal{V}$ into K sets $C_1, \ldots, C_K$, with $|C_k| = n_k$ for $k \in \{1, \ldots, K\}$. Here, K acts as a hyperparameter, that can differ from the actual number of communities eventually used for the community detection downstream evaluation task (i.e., the number of clusters in the k-means operated on the final vectors z_i). A concrete example of such a technique will be provided in Section 3.2.3 (together with explanations on how to select K). We simply assume its availability throughout these paragraphs.

We propose to leverage such an initial partition as a *prior node clustering signal* from which the GAE/VGAE encoder should benefit, but also have the ability to deviate during training, when learning the embedding space. Specifically, we propose to replace the standard input adjacency matrix A by:

$$A + \lambda A_c, \tag{13}$$

where $\lambda \geq 0$ is a scalar hyperparameter, and where A_c is the community membership matrix defined as follows:

Definition 1. Let us consider a partition of the node set $\mathcal{V}$ into K sets $C_1, \ldots, C_K$. The corresponding *community membership matrix* is defined as:

$$A_c = MM^T - I_n, \tag{14}$$

with $M \in \{0,1\}^{n \times K}$ denoting the $n \times K$ matrix where elements $M_{ik} = 1$ if and only if $i \in C_k$ according to the prior clustering.

We interpret A_c as the adjacency matrix of an alternative graph in which each cluster of our prior partition is represented by a fully connected graph, without self-loops. Since nodes are only allocated to one cluster, there exists a node ordering such that the matrix A_c is block-diagonal. In essence, $A + \lambda A_c$ aims to capture refined node similarities, by simultaneously considering some *local* information from direct neighborhoods, and some *global* information from prior node communities. The hyperparameter λ helps to balance these two aspects. In particular, setting $\lambda = 0$ results in the standard adjacency matrix.

3.2.2. From Message Passing Operators to Encoding Schemes

At first glance, $A + \lambda A_c$ could straightforwardly be incorporated as a refined message passing operator in popular GAE and VGAE encoders. For instance, one could consider its direct incorporation in:

- variants of *2-layer GCN encoders*, initially proposed by Kipf and Welling [11], as this neural architecture remains the most popular GAE/VGAE encoder in the literature [12, 15, 18, 19, 20, 21, 22, 24, 28]. Specifically, one could consider:

  - a version incorporating $A + \lambda A_c$ in both layers. Then, for example the GAE formulation[3] in Equation (2) becomes, $Z = \text{GCN}^{(1)}(A + \lambda A_c, X) = \mathcal{F}_{GCN}(A + \lambda A_c)\text{ReLU}(\mathcal{F}_{GCN}(A + \lambda A_c)XW^{(0)})W^{(1)}$.
  - a version incorporating the prior communities only on the first layer, i.e., $Z = \text{GCN}^{(2)}(A + \lambda A_c, X) = \mathcal{F}_{GCN}(A)\text{ReLU}(\mathcal{F}_{GCN}(A + \lambda A_c)XW^{(0)})W^{(1)}$.
- or, a variant of the *linear encoder*[4] proposed by Salha et al. [27]. Indeed, this simplified one-hop model without activation reached competitive performances w.r.t. multi-layer GCNs for GAE/VGAE-based community detection in recent studies [23, 27]. In this case: $Z = \text{Linear}(A + \lambda A_c, X) = \mathcal{F}_{GCN}(A + \lambda A_c)XW^{(0)}$.

However, the computational cost of evaluating each layer of a GCN or a linear encoder depends linearly on the number of edges $|\mathcal{E}| = m$ in the message passing operator [27, 30]. As the graph represented by $A + \lambda A_c$ contains at least $\sum_{k=1}^{K} n_k^2$ edges, such a direct incorporation of $A + \lambda A_c$ in encoders could incur a large computational expense.

To alleviate this cost, we will instead consider a *s-regular sparsification of* A_c, denoted by A_s in the following. In A_s, each node $i \in C_k$ is only connected to $s < n_k$ randomly selected nodes in C_k (instead of all other nodes in C_k). Therefore, the $A + \lambda A_s$ message passing operator still contains some of the prior clustering information without necessarily incurring the cost implied by the use of A_c. In particular, selecting $s \approx \frac{m}{n}$ ensures that $A + \lambda A_s$ has $O(2m)$ non-null elements, preserving the linear complexity w.r.t. m of the aforementioned encoders (in our experiments, the optimal value of s will be selected for each graph as described in Section 4.1.3). Note that we only sample A_s once at the beginning of the model training and then keep it fixed throughout training and testing. To sum up, in our upcoming experiments in Section 4 we will instead consider the following two[5] encoding schemes:

- $Z = \text{GCN}^{(2)}(A + \lambda A_s, X) = \mathcal{F}_{GCN}(A)\text{ReLU}(\mathcal{F}_{GCN}(A + \lambda A_s)XW^{(0)})W^{(1)}$.
- $Z = \text{Linear}(A + \lambda A_s, X) = \mathcal{F}_{GCN}(A + \lambda A_s)XW^{(0)}$.

In these encoders, our altered message passing scheme allows practitioners to incorporate information from prior communities in the resulting node embedding space. A given node $i \in \mathcal{C}_k \subset \mathcal{V}$, for $k \in \{1, ..., K\}$, will aggregate information from its direct neighbors and from some nodes in $\mathcal{C}_k$. By design, i will thus have an embedding vector z_i more similar to the embedding vectors of the other nodes in $\mathcal{C}_k$ than would be the case for the standard encoders based on $\mathcal{F}_{GCN}(A)$. We recall that the choice of linear and 2-layer GCN encoders is made without loss of generality. $A + \lambda A_s$ could be incorporated into other encoders including deeper GCNs, ChebNets [35] or Graph Attention Networks [36].

[3] For clarity of exposition we discuss the deterministic GAE framework (Section 2.1). However, the changes are equally applicable to the VGAE framework (Section 2.2), for which Z has to be replaced by μ and $\log \sigma$ as in Equation (7).

[4] A "*linear encoder*" is actually a particular case of GCN with a single layer and without activation function. For consistency with previous works [27, 67, 68, 69], we nonetheless adopt the "*linear encoder*" naming in this work, and use "*GCN*" to refer to the above *multi-layer* graph convolutional networks.

[5] We will favor $\text{GCN}^{(2)}$ over $\text{GCN}^{(1)}$ in the remainder of this work, as the former outperformed the latter in our experiments. To simplify the notation $\text{GCN}^{(2)}$ will be referred to as GCN in experiments.

The remainder of this Section 3.2 on encoders is organized as follows. In Section 3.2.3, we now detail how we derive the matrix A_c (that has loosely been assumed to be "available" so far) in our work. Then, in Section 3.2.4, we provide a theoretical analysis of our novel encoding strategy. It notably aims to better understand our newly introduced operators A_c and A_s in terms of the spectral filtering they induce, as well as to assess the impact of the s-regular sparsification of A_c.

3.2.3. Learning A_c and A_s with Modularity-Based Clustering

So far, for pedagogical purposes, we loosely assumed the availability of the A_c and A_s prior community membership matrices. In practice, how these matrices are *learned* plays an important role, as the empirical performance of our strategy will directly depend on the quality of the underlying prior node clusters. Throughout this paper, we will rely on *modularity* concepts to learn A_c – hence the name *Modularity-Aware GAE and VGAE*. More specifically, we will leverage the popular *Louvain* algorithm [7].

In the absence of node feature information, the Louvain greedy algorithm remains a popular and powerful approach for community detection [7]. It iteratively aims to maximize the *modularity* value [70], defined as follows:

Definition 2. Let us consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix A and nodes $i \in \mathcal{V}$ of degree $d_i = \sum_{j=1}^{n} A_{ij}$. We denote a partition of these nodes into $K \leq |\mathcal{V}|$ communities by $\{C_1, ..., C_K\}$. Then, the *modularity* associated to this partition is:

$$Q = \frac{1}{2m} \sum_{i,j=1}^{n} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(i,j), \tag{15}$$

where m is the sum of all edge weights in the graph (i.e., the number of edges for unweighted graphs), and where $\delta(i,j) = 1$ if nodes i and j belong to the same community and 0 otherwise.

In essence, the modularity compares the density of connections inside communities to connections between communities. More specifically, Equation (15) returns a scalar Q in the range $[-\frac{1}{2}, 1]$, that measures the difference between the observed fraction of (potentially weighted) edges that occur within the same community and the expected fraction of edges in a configuration model graph, which matches our observed degree distribution but allocates edges randomly without any specified community structure.

In the Louvain greedy algorithm, aiming to maximize the modularity of a graph over the set of possible cluster assignments, each node is initialized in its own community. Then, the algorithm iteratively completes two phases:

- In phase 1, for each node $i \in \mathcal{V}$, one computes the change in modularity resulting from the allocation of node i to the community of each of its neighbors. Then, i is either placed into the community leading to the greatest modularity increase, or remains in its original group if no increase is possible;
- In phase 2, a new graph is constructed. Nodes correspond to communities obtained in phase 1, and edges are formed by summing edge weights occurring between communities. Edges within a community are represented by self-loops in this new graph. One repeats phase 1 on this new graph until no further modularity improvement is possible.

Why using the Louvain method? . Our justification for the use of this method to derive A_c is threefold.

- First and foremost, it automatically selects the relevant number of prior communities K, by iteratively maximizing the modularity value.
- Secondly, it runs in $O(n \log n)$ time [7], with $n = |\mathcal{V}|$. Therefore, it scales to large graphs with millions of nodes, such as those in our experiments in Section 4.
- Thirdly, such a modularity criterion complements the encoding-decoding paradigm of standard GAE and VGAE. We argue that learning node embedding spaces from complementary criteria is beneficial. Our experiments will confirm that leveraging prior modularity-based node clusters in the GAE/VGAE outperforms the individual use of the Louvain or of the GAE/VGAE *alone*.

Note that, the use of the Louvain method is made without loss of generality as our framework remains valid for alternative graph mining methods deriving A_c and A_s.

3.2.4. Theoretical Analysis of the Encoder's Message Passing Operator

We now conduct a theoretical analysis of our newly introduced message passing operator, which we begin by motivating the spectral analysis of the matrices involved. Recall, from Equation (2), that the computations performed by a GCN at a given layer are the following,

$$\text{ReLU}(\mathcal{F}_{GCN}(A)H^{(l-1)}W^{(l-1)}). \tag{16}$$

If we consider the spectral decomposition of the message passing operator that is used in Equation (16), $\mathcal{F}_{GCN}(A) = U\Theta U^T$, where $U = [u_1, \ldots, u_n]^T$ denotes the matrix containing the eigenvectors u_i of $\mathcal{F}_{GCN}(A)$ and Θ is a diagonal matrix containing the eigenvalues θ_i of $\mathcal{F}_{GCN}(A)$. Then, the computation performed in Equation (16) can be reformulated to be,

$$\text{ReLU}(U\Theta U^T H^{(l-1)}W^{(l-1)}) = \text{ReLU}\left(\sum_{i=1}^{n} \theta_i u_i u_i^T H^{(l-1)}W^{(l-1)}\right). \tag{17}$$

Therefore, performing one message passing step of the hidden states H on a graph given by $\mathcal{F}_{GCN}(A)$, i.e., $\mathcal{F}_{GCN}(A)H^{(l-1)}$, can be interpreted as a Fourier transform of H, called *graph Fourier transform* [71], where the eigenvectors of $\mathcal{F}_{GCN}(A)$ act as a Fourier basis and the eigenvalues of $\mathcal{F}_{GCN}(A)$ define the Fourier coefficients.

When trying to perform a theoretical analysis of the message passing step in Equation (16) it often turns out to be more insightful to consider Equation (17) instead and analyze the eigenvalues and eigenvectors of the used message passing operator. Such a spectral perspective has given rise to a variety of architectures proposing learnable functions applied to the diagonal terms of Θ [72, 35, 73]. Historically, the study of spectral graph theory [74, 75], and in particular the area of graph signal processing [76, 77], has yielded much insight in the study of graphs and therefore it is somewhat unsurprising that also in the study of the GNNs the spectral analysis of these architectures is a promising avenue of analysis [78, 79, 66].

We, therefore, now provide spectral results allowing us to gain a better understanding of our proposed message passing operator and compare our proposed message passing

operator to the standard message passing operators. To characterize the eigenvectors of our newly introduced $\mathcal{F}_{GCN}(A_c)$ we rely on the concept of 2-sparse eigenvectors.

Definition 3. [80] The entries of *2-sparse eigenvectors* are all equal to 0 except for the i^{th} and j^{th} entry which equal to 1 and -1, where i and j denote two nodes which share all their neighbors, i.e., $A_{ih} = A_{jh}$ for $h \in \{1, \dots, n\} \backslash \{i, j\}$.

An extended discussion of the literature related to such 2-sparse eigenvectors and their corresponding vertices, which are sometimes referred to as twin vertices, can be found in Lutzeyer [81, p.48-9]. We are now able to characterize the spectrum and eigenvectors of $\mathcal{F}_{GCN}(A_c)$.

Proposition 4. The matrix $\mathcal{F}_{GCN}(A_c)$ has eigenvalues $\{\{1\}^K, \{0\}^{n-K}\}$, where we denote the multiset containing a given element x, y times, by $\{x\}^y$. Each non-zero eigenvalue has an associated eigenvector v_k, with $k \in \{1, \dots, K\}$, with entries $(v_k)_i = 1$ for $i \in C_k$ and $(v_k)_i = 0$ for $i \notin C_k$. The eigenspace corresponding to the zero eigenvalue has dimension $n - K$ and is spanned by, for example, a set of two-sparse eigenvectors on each of the connected components in the graph.

The proof of Proposition 4 can be found in Appendix A.1. The informal take-away from Proposition 4 is that *the cluster membership of nodes is encoded clearly and compactly in the spectrum and eigenvectors of* $\mathcal{F}_{GCN}(A_c)$. More formally, in Proposition 4 we observe that in the spectral domain the operator $\mathcal{F}_{GCN}(A_c)$, which we introduce to the encoder's message passing scheme, directly encodes the cluster membership of the different nodes and all other signals are filtered out by the 0 eigenvalues. Also in the graph domain the matrix $\mathcal{F}_{GCN}(A_c)$ clearly encodes the cluster structure by representing each cluster by a fully connected component of the graph. Therefore, the matrix $\mathcal{F}_{GCN}(A_c)$ is an appropriate choice to introduce cluster information into the message passing scheme and does so clearly in both the graph and spectral domains.

In general, the spectral filtering performed by our message passing operator, $\mathcal{F}_{GCN}(A + \lambda A_c)$, and the standard message passing operator, $\mathcal{F}_{GCN}(A)$, are different. $\mathcal{F}_{GCN}(A+\lambda A_c)$ accounts for the clustering information which we introduce. In the following theorem we provide a result that allows us to establish under which conditions the spectral filtering performed by $\mathcal{F}_{GCN}(A)$ and $\mathcal{F}_{GCN}(A + \lambda A_c)$ are equal, to gain a better understanding of the action of $\mathcal{F}_{GCN}(A + \lambda A_c)$.

Proposition 5. If $\mathcal{G}$ is composed of regular connected components, i.e., connected components containing only vertices of equal degree, and the partition of the node set defining these regular components equals the partition defining A_c, then the matrices $\mathcal{F}_{GCN}(A_c)$ and $\mathcal{F}_{GCN}(A + \lambda A_c)$ have a shared set of eigenvectors and the spectrum of $\mathcal{F}_{GCN}(A + \lambda A_c)$, denoted by $\mathcal{S}(\mathcal{F}_{GCN}(A + \lambda A_c))$, can be expressed in terms of the eigenvalues of $\mathcal{F}_{GCN}(A)$ and $\mathcal{F}_{GCN}(A_c)$, denoted by θ and η, respectively, as follows,

$$\mathcal{S}(\mathcal{F}_{GCN}(A + \lambda A_c)) = \{g_1(\theta_1) + g_2(\eta_{s(1)}), \dots, g_1(\theta_n) + g_2(\eta_{s(n)})\},$$

for affine functions g_1, g_2 parametrised by the node degrees and some permutation $s(\cdot)$ defined on the set $\{1, \dots, n\}$.

The proof of Proposition 5 can be found in Appendix A.2. Hence, we observe that for graphs consisting of regular connected components the spectral filtering performed by

our proposed message passing operator $\mathcal{F}_{GCN}(A + \lambda A_c)$ is equal to that of the standard operator $\mathcal{F}_{GCN}(A)$. For graphs consisting of regular connected components the clustering information is already contained in the spectrum of $\mathcal{F}_{GCN}(A)$ and therefore its further addition does not affect the eigenvectors of our proposed message passing operator.

Note that, in general, the spectrum of the sum of two matrices cannot be characterized by the individual spectra of the two matrices, meaning that, in general, there does not exist an exact relation between the spectra of $\mathcal{F}_{GCN}(A)$ and $\mathcal{F}_{GCN}(A_c)$ to $\mathcal{F}_{GCN}(A + \lambda A_c)$. We can however, make direct use of existing results such as Weyl's inequality [82, 83] and the extended Davis–Kahan theorem [84], which, respectively, upper bound the distance of the eigenvalues and spaces spanned by the eigenvectors of the sum of matrices and the individual matrices.

***s*-regular sparsification of our message passing operator.** We now turn to the analysis of our sparsified message passing operator A_s, which, as we will see now, still contains the external cluster information without incurring the large computation cost implied by the use of A_c.

Proposition 6. If the partition defining the connected components of A_s is a refinement of the partition defining the components of A_c, then the multiplicity of the largest eigenvalue of $\mathcal{F}_{GCN}(A_s)$ is greater or equal to the multiplicity of the largest eigenvalue of $\mathcal{F}_{GCN}(A_c)$. Further, the largest eigenvalue of both $\mathcal{F}_{GCN}(A_s)$ and $\mathcal{F}_{GCN}(A_c)$ equals 1 and the eigenvectors corresponding to the eigenvalue 1 of $\mathcal{F}_{GCN}(A_c)$ are also eigenvectors corresponding to the eigenvalue 1 of $\mathcal{F}_{GCN}(A_s)$.

The proof of Proposition 6 can be found in Appendix A.3. Informally, Proposition 6 can be interpreted to show that *the sparsification of A_c producing A_s does not impact the "informative" part of the spectrum.* Recall, that the eigenvectors corresponding to the largest eigenvalue of $\mathcal{F}_{GCN}(A_c)$ and $\mathcal{F}_{GCN}(A_s)$ are indicator vectors of our introduced cluster membership. Since the remaining eigenvectors are orthogonal to these indicator vectors we know that none of them encode our cluster membership as compactly as the eigenvectors corresponding to the largest eigenvalue.

For $\mathcal{F}_{GCN}(A_s)$ the eigenvalues corresponding to the less informative eigenvectors correspond to nonzero eigenvalues in general and we expect the choice of s to influence the impact of this uninformative part of the spectrum. This uninformative part of the spectrum can be upper bounded by adapting the bound in Friedman [85] to our message passing operator. However, in our work we choose a more practice-oriented approach by treating s as a hyperparameter of our model and find its optimal values using the procedure which is described in the upcoming Section 3.3.

3.3. Decoding and Training Strategies

So far, our work mainly considered improvements of the encoder's *architecture*. While this aspect is crucial, we also argue that previously proposed models were often *optimized* in a fashion that, by design, favors link prediction over community detection. With this in mind, this Section 3.3 now complements our contributions from Section 3.2 with revised training and optimization strategies.

3.3.1. Modularity-Inspired Losses for GAE and VGAE

As explained in Section 2, neural weight matrices of previous GAE and VGAE encoders were tuned by optimizing *reconstruction losses*, capturing the similarity between the decoded graph and the original one. Usually, these losses directly evaluate the quality of reconstructed node pairs $\hat{A}_{ij}$ w.r.t. their ground-truth counterpart A_{ij}. This includes the cross-entropy loss $\mathcal{L}_{\text{GAE}}$ from Equation (6) and the ELBO loss $\mathcal{L}_{\text{VGAE}}$ from Equation (11). We argue that this optimization strategy also contributes to explaining the underwhelming performance of some GAE and VGAE models on community detection tasks.

- By design, existing optimization strategies favor good performances on link prediction tasks, that precisely consist in accurately reconstructing connected/unconnected node pairs. However, some recent studies emphasized that a good reconstruction of *local* pairwise connections does not always imply a good reconstruction of the *global* community structure from the graph under consideration [86, 87]. This motivates the need for a revised loss function capturing some global community information.
- Besides, GAE/VGAE-based community detection experiments often consisted in running k-means algorithms in the final node embedding space (and, as stated at the beginning of Section 3.2, we also adopt this strategy). However, this results in clustering embedding vectors based on their L_2 *distances* $\|z_i - z_j\|_2$, whereas the aforementioned reconstruction losses instead often involve *inner products* ($\hat{A}_{ij} = \sigma(z_i^T z_j)$). There is thus a discrepancy between the criterion ultimately used for k-means clustering, and the one used during training to assess node similarities.

To address these issues, we propose to complement standard GAE and VGAE losses with an additional loss term, involving L_2 distances and inspired by the *modularity*[6] in Equation (15). In the case of the GAE, we will iteratively minimize by gradient descent:

$$\tilde{\mathcal{L}}_{\text{GAE}} = \mathcal{L}_{\text{GAE}} - \frac{\beta}{2m} \sum_{i,j=1}^{n} \left[A_{ij} - \frac{d_i d_j}{2m} \right] e^{-\gamma \|z_i - z_j\|_2^2}, \tag{18}$$

with hyperparameters $\beta \geq 0$, $\gamma \geq 0$. Also, for VGAE we will maximize[7]:

$$\tilde{\mathcal{L}}_{\text{VGAE}} = \mathcal{L}_{\text{VGAE}} + \frac{\beta}{2m} \sum_{i,j=1}^{n} \left[A_{ij} - \frac{d_i d_j}{2m} \right] e^{-\gamma \|z_i - z_j\|_2^2}. \tag{19}$$

We note that $\tilde{\mathcal{L}}_{\text{GAE}}$ and $\tilde{\mathcal{L}}_{\text{VGAE}}$ are independent of the ground-truth community information. Consistently with previous efforts on community detection using GAE and VGAE (see Section 2.4), we only use ground-truth communities for the final *evaluation* of embedding vectors – not to learn these vectors.

In Equations (18) and (19), the exponential term (taking values in $[0, 1]$) acts as a *soft counterpart of the common community indicator* $\delta(i, j) \in \{0, 1\}$ in Equation (15). It tends to 1 when nodes i and j get closer in the embedding space, and tends to 0 when they move apart.

[6] We emphasize that this new term does *not* involve the prior Louvain clusters used in A_c.

[7] We recall that $\mathcal{L}_{\text{GAE}}$ is *minimized* while $\mathcal{L}_{\text{VGAE}}$ is *maximized*, hence the occurrence of a *minus* term in Equation (18) but a *plus* term in Equation (19).

In essence, we expect the addition of such a *global regularizer* to $\mathcal{L}_{\text{GAE}}$ and $\mathcal{L}_{\text{VGAE}}$ to encourage closer embedding vectors (in the L_2 distance) of densely connected parts of the original graph, and therefore to permit a k-means-based *detection of communities with higher modularity values*. On the other hand, the remaining presence of the original $\mathcal{L}_{\text{GAE}}$ or $\mathcal{L}_{\text{VGAE}}$ term[8] in the loss aims to *preserve good performances on link prediction*. The hyperparameter β balances the relative importance of the modularity regularizer w.r.t. the pairwise node pairs reconstruction loss, while the hyperparameter γ regulates the magnitude of $\|z_i - z_j\|_2^2$ in the exponential term. Our experiments will show that proper tuning of β and γ permits us to improve community detection while jointly preserving performances on link prediction.

The use of a modularity-inspired regularizer in the loss of the *Modularity-Aware GAE and VGAE* builds upon several studies, which were not studying the GAE/VGAE frameworks but emphasized the benefits of various modularity-inspired losses for learning community-preserving node embedding representations [86, 88, 89]. In our setting, we favor the use of a *soft* modularity instead of the term in Equation (15), as it permits 1) to obtain a differentiable loss, and 2) to avoid the actual reconstruction of node communities at each training iteration, which would incur a larger computational expense.

To conclude we note that, as for a complete evaluation of $\mathcal{L}_{\text{GAE}}$ and $\mathcal{L}_{\text{VGAE}}$, computing the modularity-inspired terms in Equations (18) and (19) *on the entire graph* would be of quadratic complexity w.r.t. the number of nodes in the graph. In some of our experiments where such a complexity would be unaffordable (roughly, when $n \geq 50\ 000$), we will rely on the FastGAE method [24] to approximate modularity-inspired terms. This method was already mentioned in Section 2.1.3, to approximate $\mathcal{L}_{\text{GAE}}$ and $\mathcal{L}_{\text{VGAE}}$ losses on large graphs by computing them only on strategically-selected random sampled *subgraphs* of $O(n)$ size, drawn at each training iteration. We will subsequently approximate modularity-inspired terms on the same subgraphs. This ensures a linear complexity for each evaluation of our proposed $\tilde{\mathcal{L}}_{\text{GAE}}$ and $\tilde{\mathcal{L}}_{\text{VGAE}}$ losses in Equations (18) and (19), respectively. As our revised encoders from Section 3.2 also exhibit a linear complexity w.r.t. n, our whole *Modularity-Aware GAE and VGAE* are scalable to graphs with hundreds of thousands to millions of nodes.

3.3.2. On the Selection of Hyperparameters

We expect our modularity-inspired losses to improve the training of our linear/GCN encoders for community detection, i.e., the tuning of their *weight matrices*. However, in addition to these weight matrices, our *Modularity-Aware GAE and VGAE* involve several other hyperparameters, that also play a key role. This includes the standard hyperparameters of GAE and VGAE models (e.g., the number of training iterations, the learning rate, the dimensions of encoding layers and, potentially, the dropout rate [90]), but also our newly introduced hyperparameters: λ and s from our encoders, as well as β and γ from our losses.

In previous research, the selection procedure for such important hyperparameters was sometimes solely based on the optimization of AUC or AP scores on link prediction

[8] Our experiments will consider the original $\mathcal{L}_{\text{GAE}}$ or $\mathcal{L}_{\text{VGAE}}$ from Equations (6) and (11) and originally formulated by Kipf and Welling [11]. Nonetheless, one can observe that our modularity-inspired global regularizer term could be optimized *in conjunction with other reconstruction losses*. For instance, modularity-inspired terms could be added to the variant formulation of ELBO loss from Choong et al. [12], that incorporates *Gaussian mixtures* in the Kullback-Leibler divergence.

validation sets [17, 24], following the train/validation/test splitting procedure initially adopted by Kipf and Welling [11] and previously described in Section 2.3.1. However, intuitively, the best hyperparameters for community detection might differ from the best ones for link prediction. Such a selection procedure might therefore be suboptimal for community detection problems.

To tackle this issue, and to complement our novel encoders (Section 3.2) and losses (Section 3.3.1), we propose an alternative hyperparameter selection procedure w.r.t. previous practices. As community detection is an *unsupervised* downstream task, we cannot rely on train/validation/test splits as for the *supervised* link prediction binary classification task[9]. Consistent with our already described contributions, we rather propose to rely on *modularity* scores, as it is an unsupervised criterion computed independently of the unobserved ground-truth clusters. More precisely, to select relevant hyperparameters, we will:

- firstly, construct link prediction train/validation/test sets, as in Section 2.3.1;
- then, select hyperparameters that maximize the average of:
  - the *AUC score* obtained for link prediction on the validation set;
  - the *modularity score* Q defined in Equation (15). This score is obtained from the communities extracted by running a k-means on the final vectors z_i, learned from the train graph (all nodes are visible but some edges - the validation and test ones - are masked).

We expect this dual criterion to facilitate the identification of hyperparameters that will be jointly relevant for link prediction and community detection downstream applications.

4. Experimental Evaluation

We now present an in-depth experimental evaluation of our proposed *Modularity-Aware GAE and VGAE* models together with relevant baselines. In Section 4.1 we first describe our experimental setting. Then in Section 4.2, we report and discuss our results.

4.1. ***Experimental Setting***

4.1.1. Datasets

In the following, we provide an experimental evaluation on seven graphs of various origins, characteristics, and sizes. Their statistics are summarized in Table 3.

First and foremost, we consider the *Cora, Citeseer and Pubmed citation networks* [30]. We study two versions of each of these datasets, *with* and *without* node features that correspond to bag-of-words vectors of dimensions $f = 1433$, 3703 and 500, respectively. In these datasets, nodes are clustered in 6, 7 and 3 topic classes, respectively, that will act as the communities to be detected in our experiments. These three citations networks are by far the most commonly used graph datasets to evaluate GAE and VGAE models

[9] We recall that the ground-truth communities of each node will be *unavailable* during training. They will only be ultimately revealed for model evaluation, to compare the agreement of the node partition proposed by our GAE or VGAE model to the ground-truth partition.

Table 3: Statistics of graph datasets

Dataset	Number of nodes	Number of edges	Number of communities
Blogs	1 224	19 025	2
Cora	2 708	5 429	6
Citeseer	3 327	4 732	7
Pubmed	19 717	44 338	3
Cora-Large	23 166	91 500	70
SBM	100 000	1 498 844	100
Deezer-Album	2 503 985	25 039 155	20

[10, 11, 12, 16, 17, 19, 20, 21, 22, 23, 28, 29]. We therefore see value in studying them as well, especially in their *featureless* version where, as explained in Section 2.4.4, previous GAE and VGAE extensions fall short on community detection.

We complete our experimental evaluation with four other datasets. Firstly, we consider the ten times larger version of Cora used in [27] for community detection, and referred to as *Cora-Large* in the following. Nodes are documents clustered in 70 topic-related communities. Additionally, we consider the *Blogs web graph* also used in [27], where nodes correspond to webpages of political blogs connected through hyperlinks. The blogs are clustered in two communities corresponding to politically left-leaning or right-leaning blogs. Thirdly, as in Salha et al. [24], we examine a graph generated from a *stochastic block model*, which is a generative model for community-based random graphs [91]. We follow the parameterization of Salha et al. [24] and generate this graph, which we refer to as *SBM*, as follows. Nodes are clustered in 100 ground-truth communities of 1000 nodes each. Nodes from the same community are connected with probability $p = 2 \times 10^{-2}$, while nodes from different communities are connected with probability $q = 2 \times 10^{-4} < p$. Albeit being synthetic, this graph includes actual node communities by design, and is, therefore, relevant to evaluate community detection methods.

Lastly, we consider an industrial-scale private graph provided by the global music streaming service Deezer[10]. Graph-based methods are at the core of Deezer's recommender systems [55]. In the graph under consideration in this study, nodes correspond to 2.5 million *music albums* available on the service. They are connected through an undirected edge when they are regularly *co-listened* by Deezer users (as assessed by internal usage metrics computed from millions of users, but undisclosed in this work for privacy reasons). Deezer is jointly interested in 1) predicting new connections in the graph corresponding to new albums pairs that users would enjoy listening to together, which is achieved by performing the *link prediction* task; and 2) learning groups of similar albums, with the aim of providing usage-based recommendations (i.e., if users listen to several albums from a community, other unlistened albums from this same community could be recommended to them), which is achieved by performing the *community detection* task. In such an industrial application, learning high-quality album representations that would jointly enable effective link prediction and community detection would therefore be desirable. For evaluation, node communities will be compared to a ground-truth clustering of albums in 20 groups defined by their main *music genre*, allowing us to assess the musical homogeneity of the node communities proposed by each model.

[10] https://www.deezer.com/ (accessed October 13, 2021).

4.1.2. Tasks

For each of these seven graphs, we assess the performance of our models on two downstream tasks.

- **Task 1:** We first consider a pure *community detection* task, consisting in the extraction of a partition of the node set $\mathcal{V}$ which ideally agrees with the ground-truth communities of each graph. Communities will be retrieved by running the k-means algorithm (with k-means++ initialization [92]) in the final embedding space of each model to cluster the vectors z_i (with k matching the known number of communities from Table 3); except for some baseline methods that explicitly incorporate another strategy to partition nodes (see Section 4.1.3). We compare the obtained partitions to the ground-truth using the popular *Adjusted Mutual Information (AMI)* and *Adjusted Rand Index (ARI)* scores[11] for clustering evaluation.
- **Task 2:** We also consider a *joint link prediction and community detection* task. In such a setting, we learn all node embedding spaces from *incomplete* versions of the seven graphs, where 15% of edges were randomly masked. We create a validation and a test set from these masked edges (from 5% and 10% of edges, respectively, as in Kipf and Welling [11]) and the same number of randomly picked unconnected node pairs acting as "non-edge" negative pairs. Then, using decoder predictions $\hat{A}_{ij}$ computed from vectors z_i and z_j, we evaluate each model's ability to distinguish edges from non-edges, i.e., *link prediction*, from the embedding space, using the *Area Under the ROC Curve (AUC)* and *Average Precision (AP)* scores[11] on the test sets. Jointly, we also evaluate the community detection performance obtained from such incomplete graphs, using the same methodology and AMI/ARI scores as in Task 1.

In the case of Task 2, we expect AMI and ARI scores to slightly decrease w.r.t. Task 1, as models will only observe *incomplete* versions of the graphs when learning embedding spaces. Task 2 will further assess whether empirically improving community detection inevitably leads to deteriorating the original good performances of GAE and VGAE models on link prediction. As our proposed Modularity-Inspired GAE and VGAE are designed for *joint link prediction and community detection*, we expect them to 1) reach comparable (or, ideally, identical) AUC/AP link prediction scores w.r.t. standard GAE and VGAE, while 2) reaching better community detection scores.

4.1.3. Details on Models

For the aforementioned evaluation tasks and graphs, we will compare the performances of our proposed *Modularity-Aware GAE and VGAE* models to standard GAE and VGAE and to several other baselines. All results reported below will verify $d = 16$, i.e., all node embedding models will learn embedding vectors z_i of dimension 16. We also tested models with $d \in \{32, 64\}$ by including them in our grid search space and reached similar conclusions to the $d = 16$ setting (we report and further discuss the impact of d in Section 4.2. Note, the dimension d is a selectable parameter in our public implementation, permitting direct model training on any node embedding dimension).

[11] Scores are computed via scikit-learn [48], using formulas provided here: https://scikit-learn.org/stable/modules/classes.html?highlight=metrics#module-sklearn.metrics (accessed October 13, 2021).

Table 4: Complete list of optimal hyperparameters of Modularity-Aware GAE and VGAE models

Dataset	Learning rate	Number of iterations	Dropout rate	Use of FastGAE [24] (if yes: subgraphs size)	λ	β	γ	s
Blogs	0.01	200	0.0	No	0.5	0.75	2	10
Cora (featureless)	0.01	500	0.0	No	0.25	1.0	0.25	1
Cora (with features)	0.01	300	0.0	No	0.001	0.01	1	1
Citeseer (featureless)	0.01	500	0.0	No	0.75	0.5	0.5	2
Citeseer (with features)	0.01	500	0.0	No	0.75	0.5	0.5	2
Pubmed (featureless)	0.01	500	0.0	No	0.1	0.5	0.1	5
Pubmed (with features)	0.01	700	0.0	No	0.1	0.5	10	2
Cora-Large	0.01	500	0.0	No	0.001	0.1	0.1	10
SBM	0.01	300	0.0	Yes (10 000)	0.5	0.1	2	10
Deezer-Album	0.005	600	0.0	Yes (10 000)	0.25	0.25	1	5

Modularity-Aware GAE and VGAE. We trained two versions of our Modularity-Aware GAE and VGAE: one with the *linear encoder* described in Section 3.2.2, and one with the *2-layer GCN encoder* ($\text{GCN}^{(2)}$). The latter encoder includes a 32-dimensional hidden layer. We recall that link prediction is performed from inner product decoding $\hat{A}_{ij} = \sigma(z_i^T z_j)$, and that community detection is performed via a k-means on the final vectors z_i learned by each model.

During training, we used the Adam optimizer [93], without dropout (but we tested models with dropout values in $\{0, 0.1, 0.2\}$ in our grid search optimization). All hyperparameters were carefully tuned following the procedure described in Section 3.3.2. For each graph, we tested learning rates from the grid $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$, number of training iterations in $\{100, 200, 300, ..., 800\}$, with $\lambda \in \{0, 0.01, 0.05, 0.1, 0.2, 0.3, ..., 1.0\}$, $\beta \in \{0, 0.01, 0.05, 0.1, 0.25, 0.5, 1.0, 1.5, 2.0\}$, $\gamma \in \{0.1, 0.2, 0.5, 1.0, 2, 5, 10\}$ and $s \in \{1, 2, 5, 10\}$. The best hyperparameters for each graph are reported in Table 4. We adopted the same optimal hyperparameters for GAE *and* VGAE variants (a result which is consistent with the literature [11]). Lastly, as exact loss computation was computationally unaffordable for our two largest graphs, SBM and Deezer-Album, their corresponding models were trained by using the FastGAE method [24], approximating losses by reconstructing degree-based sampled subgraphs of $n = 10\,000$ nodes (a different one at each training iteration).

We used Tensorflow [94], training our models (as well as GAE/VGAE baselines described below) on an NVIDIA GTX 1080 GPU, and running other operations on a double Intel Xeon Gold 6134 CPU[12]. Along with this paper, we will publicly release our source code on GitHub for reproducibility and to encourage future usage of our method[13].

Standard GAE and VGAE. We compare the above Modularity-Aware GAE and VGAE to two variants of the standard GAE and VGAE: one with 2-layer GCN encoders with 32-dimensional hidden layer (which is equal to the seminal GAE and VGAE from Kipf and Welling [11]) and one with a linear encoder (which equals the linear GAE and VGAE

[12]On our machines, running times of the Modularity-Aware GAE and VGAE models were comparable to running times of their standard GAE and VGAE counterparts. For example, training each variant of VGAE on the Pubmed graph for 500 training iterations and with $s = 5$ approximately takes 25 minutes on a single GPU (without the FastGAE method, which significantly speeds up training [24]). This is consistent with our claims on the comparable complexity of Modularity-Aware and standard models.

[13]https://github.com/GuillaumeSalhaGalvan/modularity_aware_gae

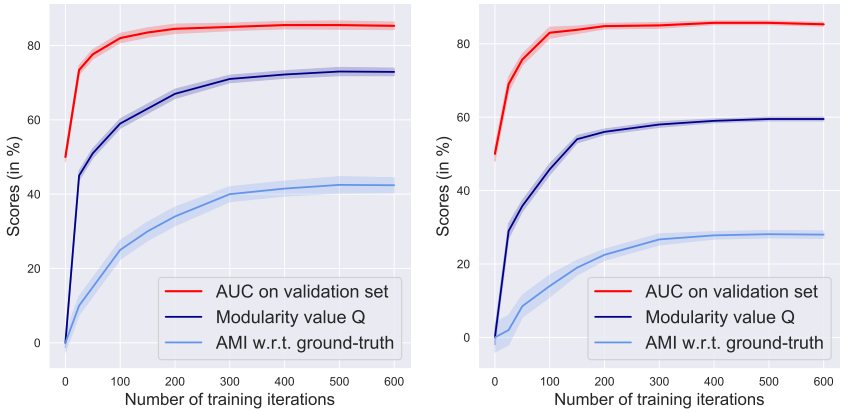
from Salha et al. [27]). We note that these are particular cases of our Modularity-Aware GAE/VGAE with GCN or linear encoder and with $\lambda = 0$ and $\beta = 0$.

As for our Modularity-Aware models, link prediction is performed from inner product decoding, and community detection via a k-means on vectors z_i. We also adopt a similar model selection procedure as for our Modularity-Aware GAE and VGAE to select hyperparameters (see Section 3.3.2). We selected similar learning rates and number of iterations to the values reported in Table 4.

Other baselines. For completeness, we also compare the standard and Modularity-Aware GAE/VGAE to several other relevant baselines. First and foremost, we report experiments on the VGAECD [12] and VGAECD-OPT [23] models, designed for community detection and discussed in Section 2.4.3. We use our own Tensorflow reimplementation of these models[14]. We set similar hyperparameters to the above other GAE/VGAE-based models. In all models, the number of Gaussian mixtures matches the ground-truth number of communities of each graph. Besides, we also report experiments on the DGVAE [29] model also discussed in Section 2.4.3, setting similar learning rates and layer dimensions to the above GAE/VGAE-based models, and using the authors' public implementation. In the case of DGVAE, we use 2-layer GCN encoders for consistency with other models of our experiments; we nonetheless acknowledge that Li et al. [29] also proposed another encoding scheme, denoted Heatts in their paper (but unavailable in their public code at the time of writing) that could replace GCNs both in DGVAE and in Modularity-Aware GAE and VGAE. We also report experiments on the ARGA and ARVGA models from Pan et al. [15] that incorporate an adversarial regularization scheme, with similar hyperparameters and using the authors' implementation. ARGA and ARVGA emerged as some of the most cited GAE/VGAE extensions and, while they were not specifically introduced for community detection, Pan et al. [15] reported empirical gains on this task w.r.t. standard GAE/VGAE (on graphs with node features).

We furthermore consider three additional baselines not utilizing the autoencoder paradigm. Firstly, we report results obtained from the popular node embedding methods *node2vec* [95] and *DeepWalk* [49]. We used the authors' respective implementations, training models from 10 random walks with length 80 per node, window size of 5 and on a single epoch. For node2vec, we further set $p = q = 1$. We use a similar strategy to our aforementioned GAE/VGAE models (k-means/inner products) for community detection and link prediction from embedding spaces. Lastly, we also compare to the *Louvain* community detection method, using the authors' implementation [7]. We see value in comparing our methods to a direct use of Louvain, as this method 1) often emerged as a simple but competitive alternative to GAE/VGAE for community detection (see Section 2.4.2), and 2) is directly leveraged in our proposed *Modularity-Aware GAE/VGAE* as a pre-processing step for the computation of A_c and A_s (see Section 3.2).

[14] Authors of VGAECD/VGAECD-OPT did not release any public implementation of their models, and we were unable to reach them by e-mail. We note that we obtained some inconsistent results w.r.t. their original performances (specifically, we reached better performances on featureless graphs, and lower performances on graphs with node features), even when adopting their set of hyperparameters. For the sake of transparency, we will thereafter 1) report scores obtained through our own re-implementation, and 2) also specify scores reported in their original work, when they were significantly different. Authors followed an experimental setting and pure community detection task similar to ours.



(a) Cora (b) Pubmed

Figure 2: Identification of the required number of training iterations, for Modularity-Aware VGAE with linear encoders trained on the featureless (a) Cora, and (b) Pubmed graphs. The plots report the evolution of the modularity Q (dark blue) and AUC link prediction scores on validation sets (red) w.r.t. the number of model training iterations in gradient descent. By looking at the red curves only, one might choose to stop training models after 200 iterations as in [11], as the AUC validation scores have almost stabilized. However, the dark blue curves emphasize that Q still increases up to 400-500 training iterations for both graphs. By also using Q for hyperparameter selection (as we proposed), one will therefore continue training VGAE models up to 400-500 iterations. The light blue curves confirm that such a strategy eventually leads to better AMI final scores w.r.t. ground-truth communities. Note, that the light blue curves could *not* be directly used for tuning, as ground-truth communities are assumed to be unavailable at training time.

4.2. ***Results***

We now present our experimental results. In Section 4.2.1 we analyze the impact of our proposed hyperparameter selection procedure. In Section 4.2.2 and 4.2.3, we discuss results on Task 1 and Task 2, respectively. Finally, we mention limitations and possible extensions of our approach in Section 4.2.4.

4.2.1. On the Selection of Hyperparameters

In Section 3.3.2, we proposed an alternative hyperparameter selection procedure w.r.t. previous practices in the literature. Based on the joint maximization of AUC validation scores for link prediction and modularity scores Q, it aims to identify more relevant GAE/VGAE hyperparameters for joint link prediction and community detection. Recall, that the resulting optimal parameters are displayed in Table 4.

In our experiments, this procedure did not modify our choices of learning rates and dropout rates for the different GAE/VGAE models under consideration, w.r.t. a standard selection solely relying on AUC validation scores. It had a more noticeable impact on the choices of clustering-related hyperparameters in Modularity-Aware GAE and VGAE (i.e., λ, β, γ, and s) as well as on the required number of training iterations in the gradient descent.

Figure 2 provides an example of this phenomenon, for the number of training iterations required to train Modularity-Aware VGAE models on the featureless Cora and Pubmed

Table 5: Results for Task 1 and Task 2 on the featureless Cora graph, using Modularity-Aware GAE and VGAE with Linear and GCN encoders, their standard GAE and VGAE counterparts, and other baselines. All node embedding models learn embedding vectors of dimension $d = 16$, with other hyperparameters set as described in Section 4.1.3. Scores are averaged over 100 runs. For Task 2, link prediction results are reported from test sets (edges masked for the original graph in addition to the same number of randomly picked unconnected node pairs). **Bold** numbers correspond to the best performance for each score. Scores *in italic* are within one standard deviation range from the best score.

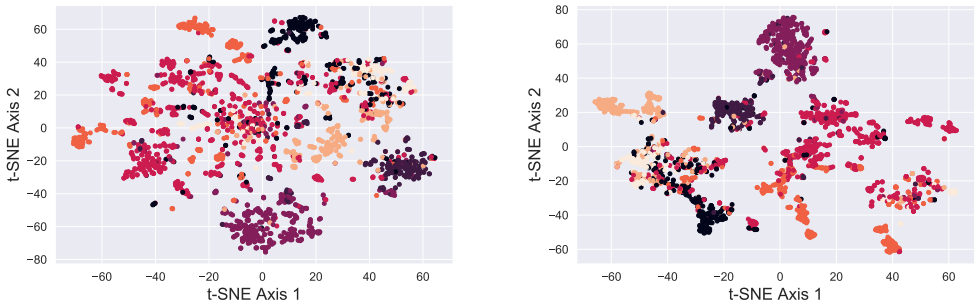
Models (Dimension $d = 16$)	**Task 1: Community Detection on complete graph**		**Task 2: Joint Link Prediction and Community Detection on graph with 15% of edges being masked**			
	AMI (in %)	**ARI (in %)**	**AMI (in %)**	**ARI (in %)**	**AUC (in %)**	**AP (in %)**
Modularity-Aware GAE/VGAE Models						
Linear Modularity-Aware VGAE	**46.65 ± 0.94**	*39.43 ± 1.15*	*42.86 ± 1.65*	*34.53 ± 1.97*	*85.96 ± 1.24*	*87.21 ± 1.39*
Linear Modularity-Aware GAE	*46.58 ± 0.40*	**39.71 ± 0.41**	**43.48 ± 1.12**	**35.51 ± 1.20**	**87.18 ± 1.05**	*88.53 ± 1.33*
GCN-based Modularity-Aware VGAE	43.25 ± 1.62	35.08 ± 1.88	41.03 ± 1.55	*33.43 ± 2.17*	84.87 ± 1.14	85.16 ± 1.23
GCN-based Modularity-Aware GAE	44.39 ± 0.85	38.70 ± 0.94	41.13 ± 1.35	*35.01 ± 1.58*	*86.90 ± 1.16*	*87.55 ± 1.26*
Standard GAE/VGAE Models						
Linear VGAE	37.12 ± 1.46	26.83 ± 1.68	32.22 ± 1.76	21.82 ± 1.80	85.69 ± 1.17	**89.12 ± 0.82**
Linear GAE	35.05 ± 2.55	24.32 ± 2.99	28.41 ± 1.68	19.45 ± 1.75	84.46 ± 1.64	*88.42 ± 1.07*
GCN-based VGAE	34.36 ± 3.66	23.98 ± 5.01	28.62 ± 2.76	19.70 ± 3.71	85.47 ± 1.18	*88.90 ± 1.11*
GCN-based GAE	35.64 ± 3.67	25.33 ± 4.06	31.30 ± 2.07	19.89 ± 3.07	85.31 ± 1.35	*88.67 ± 1.24*
Other Baselines						
Louvain	42.70 ± 0.65	24.01 ± 1.70	39.09 ± 0.73	20.19 ± 1.73	–	–
VGAECD	36.11 ± 1.07	27.15 ± 2.05	33.54 ± 1.46	24.32 ± 2.25	83.12 ± 1.11	84.68 ± 0.98
VGAECD-OPT	38.93 ± 1.21	27.61 ± 1.82	34.41 ± 1.62	24.66 ± 1.98	82.89 ± 1.20	83.70 ± 1.16
ARGVA	34.97 ± 3.01	23.29 ± 3.21	28.96 ± 2.64	19.74 ± 3.02	85.85 ± 0.87	*88.94 ± 0.72*
ARGA	35.91 ± 3.11	25.88 ± 2.89	31.61 ± 2.05	20.18 ± 2.92	85.95 ± 0.85	*89.07 ± 0.70*
DVGAE	35.02 ± 2.73	25.03 ± 4.32	30.46 ± 4.12	21.06 ± 5.06	85.58 ± 1.31	*88.77 ± 1.29*
DeepWalk	36.58 ± 1.69	27.92 ± 2.93	30.26 ± 2.32	20.24 ± 3.91	80.67 ± 1.50	80.48 ± 1.28
node2vec	41.64 ± 1.25	34.30 ± 1.92	36.25 ± 1.38	29.43 ± 2.21	82.43 ± 1.23	81.60 ± 0.91

graphs. The figure shows that, unlike our proposed procedure jointly based on AUC and Q, a hyperparameter selection based solely on AUC validation scores leads to earlier stopping of the model training and *suboptimal* performances on community detection. This reaffirms the empirical relevance of our proposed procedure, and that optimal hyperparameters for joint link prediction and community detection might differ from those for link prediction only. Moreover, we note that, while Figure 2 focuses on Modularity-Aware VGAE, our procedure also leads to the selection of a larger number of training iterations for the other GAE/VGAE-based methods under consideration in this work (values are similar to those in Table 4), which explains why, on some occasions, we will report slightly improved results w.r.t. those obtained in the original papers.

4.2.2. Results for Community Detection on Original Graphs (Task 1)

We now focus on the "pure" community detection task (Task 1), performed by models trained on graphs, where no edges are removed for model training as previously introduced in Section 4.1.2. The second and third column of Table 5 reports mean AMI and ARI scores on Cora for this task along with standard deviations over 100 runs, for Modularity-Aware GAE and VGAE models (with linear or GCN encoders), their standard counterparts and other baselines.

We draw several conclusions from Table 5. Foremost, previous conclusions [12, 17, 23, 24] on the limitations of standard GAE and VGAE for community detection are confirmed: in Table 5, these methods are notably outperformed by a direct use of the Louvain method (e.g., 42.70% vs 34.36% mean AMI scores for Louvain vs GCN-based VGAE). We also observe that previous GAE/VGAE extensions, reported as baselines,



(a) Linear Standard VGAE

(b) Linear Modularity-Aware VGAE

Figure 3: Visualization of node embedding representations for the featureless Cora graph, learned by (a) Standard VGAE, and (b) Modularity-Aware VGAE, with linear encoders. The plots were obtained using the t-SNE method for high-dimensional data visualization. Colors denote ground-truth communities, that were not available during training. Although community detection is not perfect (both methods return AMI scores $< 50\%$ in Table 5), node embedding representations from (b) provide a more visible separation of these communities. Specifically, in Table 5, using Linear Modularity-Aware VGAE for community detection leads to an increase of 9 AMI points (Task 1) to 10 AMI points (Task 2) for community detection w.r.t. Linear Standard VGAE, while preserving comparable performances on link prediction (Task 2).

actually provide few empirical benefits w.r.t. standard GAE and VGAE for this *featureless* graph (e.g., only +1.81 AMI points for VGAECD-OPT[15] vs Linear VGAE). Such a result, in conjunction with the improved performances of these same baselines on graphs *with features* (see thereafter), tends to confirm our initial diagnosis that various GAE/VGAE extensions for community detection mainly benefit from the presence of node features.

On the contrary, our proposed Modularity-Aware GAE and VGAE models, incorporating Louvain clusters as a prior signal in the GAE's and VGAE's encoders, significantly outperform both the use of the Louvain method alone, and the use of GAE and VGAE alone (e.g., with a top 46.65% mean AMI for Modularity-Aware VGAE with linear encoders, and a top 39.71% mean ARI score for Modularity-Aware GAE with linear encoders). Modularity-Aware models also compare favorably to the baselines under consideration (e.g., with +12.1 ARI points for Linear Modularity-Aware GAE w.r.t. VGAECD-OPT), while also providing less volatile results w.r.t. standard GAE/VGAE. Furthermore, we note that Modularity-Aware models with linear encoders tend to outperform their GCN-based counterparts (consistently with previous findings from [27, 23] on Cora) and that GAE and VGAE reach comparable scores. In addition to these results, Figure 3 visualizes the node embeddings learned by our models using t-SNE[16] [96].

Overall, we obtain similar conclusions on the other graph datasets. Following the format of Table 5, columns two and three of Table 6 present detailed community detection

[15]The increase is even smaller when replacing AMI scores obtained via our re-implementation of VGAECD and VGAECD-OPT (i.e., 36.11% and 38.93%, respectively) by AMI scores originally reported in [23] for these methods (i.e., 28.22% and 37.35%, respectively), which are lower than ours.

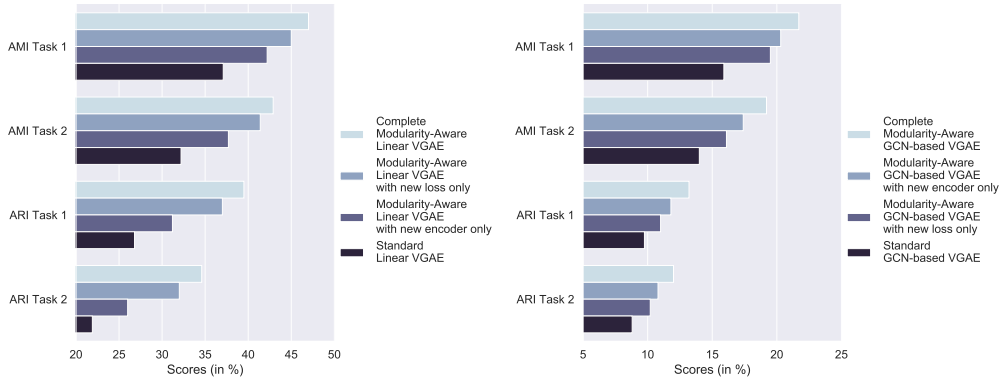
[16]We used the scikit-learn [48] implementation of this data visualization method: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html (accessed October 13, 2021).

Table 6: Results for Task 1 and Task 2 on the featureless Pubmed graph, using Modularity-Aware GAE and VGAE with Linear and GCN encoders, their standard GAE and VGAE counterparts, and other baselines. All node embedding models learn embedding vectors of dimension $d = 16$, with other hyperparameters set as described in Section 4.1.3. Scores are averaged over 100 runs. For Task 2, link prediction results are reported from test sets (edges masked during training + same number of randomly picked unconnected node pairs). **Bold** numbers correspond to the best performance for each score. Scores *in italic* are within one standard deviation range from the best score.

Models (Dimension $d = 16$)	**Task 1: Community Detection on complete graph**		**Task 2: Joint Link Prediction and Community Detection on graph with 15% of edges being masked**			
	AMI (in %)	**ARI (in %)**	**AMI (in %)**	**ARI (in %)**	**AUC (in %)**	**AP (in %)**
Modularity-Aware GAE/VGAE Models						
Linear Modularity-Aware VGAE	28.12 ± 0.29	29.01 ± 0.51	25.93 ± 0.65	23.76 ± 0.49	**85.76 ± 0.37**	87.77 ± 0.31
Linear Modularity-Aware GAE	*28.54 ± 0.24*	26.36 ± 0.34	**26.38 ± 0.43**	21.30 ± 0.59	84.39 ± 0.32	*87.92 ± 0.40*
GCN-based Modularity-Aware VGAE	28.08 ± 0.27	28.14 ± 0.33	25.70 ± 0.86	22.65 ± 0.80	84.70 ± 0.24	86.64 ± 0.15
GCN-based Modularity-Aware GAE	**28.74 ± 0.28**	26.71 ± 0.47	25.52 ± 0.45	20.52 ± 0.31	85.07 ± 0.35	*88.27 ± 0.39*
Standard GAE/VGAE Models						
Linear VGAE	22.16 ± 2.02	13.90 ± 3.47	21.78 ± 2.57	13.81 ± 3.17	84.57 ± 0.51	**88.31 ± 0.44**
Linear GAE	12.61 ± 4.61	6.37 ± 3.86	12.60 ± 4.67	6.21 ± 1.75	82.03 ± 0.32	87.71 ± 0.24
GCN-based VGAE	20.11 ± 3.05	13.12 ± 3.10	17.34 ± 2.99	8.71 ± 3.05	82.19 ± 0.88	87.51 ± 0.55
GCN-based GAE	20.12 ± 2.89	14.21 ± 2.78	16.75 ± 3.36	9.18 ± 2.71	82.33 ± 1.32	87.20 ± 0.58
Other Baselines						
Louvain	20.06 ± 0.27	10.34 ± 0.99	16.71 ± 0.46	8.32 ± 0.79	–	–
VGAECD	20.32 ± 2.95	13.54 ± 2.98	17.39 ± 3.04	9.21 ± 3.12	82.05 ± 0.90	87.30 ± 0.53
VGAECD-OPT	22.50 ± 1.99	14.58 ± 2.86	21.98 ± 2.46	15.22 ± 2.92	82.03 ± 0.82	87.41 ± 0.53
ARGVA	20.73 ± 3.10	13.94 ± 3.12	17.63 ± 3.19	9.19 ± 3.09	84.07 ± 0.55	87.73 ± 0.49
ARGA	20.98 ± 2.90	14.79 ± 2.80	17.21 ± 3.01	9.59 ± 2.76	83.73 ± 0.53	*87.90 ± 0.45*
DVGAE	23.15 ± 2.52	15.02 ± 3.33	22.10 ± 2.50	14.62 ± 2.96	83.21 ± 0.92	*88.17 ± 0.49*
DeepWalk	*28.53 ± 0.43*	29.61 ± 0.33	15.80 ± 1.05	16.16 ± 1.75	80.63 ± 0.42	81.03 ± 0.54
node2vec	*28.52 ± 1.12*	**30.63 ± 1.14**	23.88 ± 0.54	**25.90 ± 0.65**	81.03 ± 0.30	82.33 ± 0.41

results for the featureless Pubmed graph. Table 7 reports more summarized results for all other graph datasets under consideration, with and without node features (when available). While Louvain outperforms standard GAE/VGAE in 5 featureless graphs out of 7 in Table 7 (e.g., 19.81% vs 15.79% mean AMI scores for Louvain vs GCN-based VGAE on Deezer-Album), our Modularity-Aware models manage to achieve either comparable or better performances w.r.t. standard models, Louvain and other baselines in the wide majority of experiments. Furthermore, throughout Table 7, we observe that linear encoders outperform their GCN-based counterparts in 8/10 experiments, and that VGAE models outperform GAE models in 8/10 experiments (even though performances are often relatively close, as for Cora). Moreover we emphasize that, while all tables report results for fixed embedding dimensions of $d = 16$, we reached similar conclusions for $d \in \{32, 64\}$. Although performances sometimes improved by increasing d, the *ranking* of methods under consideration remained similar: for instance, by setting $d = 64$, Deepwalk's mean AMI score increased from 36.58% to roughly 41% on the featureless Cora graph, while the mean AMI score from our Linear Modularity-Aware GAE simultaneously increased from 46.58% to 47.80%. Lastly, while we observed that our results were quite sensitive to our choice of hyperparameters λ, β, and γ, no direct link with the number of nodes, of edges, or of ground-truth communities in the graph seems to emerge from our experiments.

Interestingly, we also observe that combining the Louvain method and GAE/VGAE in our Modularity-Aware models might be empirically beneficial *even when standard GAE/VGAE initially outperform the Louvain method*. For instance in Table 6, our Linear Modularity-Aware VGAE outperforms Linear Standard VGAE (e.g., with 28.12% vs



(a) Cora (b) Deezer

Figure 4: Comparison of two "complete" Modularity-Aware VGAE, trained on (a) featureless Cora and (b) Deezer-Album with variants of these models only leveraging the new *encoder* from Section 3.2, or the new *loss* from Section 3.3. We observe that incorporating any of these two components improves community detection on these two graphs w.r.t. Standard VGAE. Moreover, using both components *simultaneously* leads to the best results. Note, the optimal pair (λ, β) for complete models might differ from the optimal λ (resp. β) when incorporating the new encoder (resp. loss) only.

22.16% mean AMI scores), despite the fact that this standard model initially outperformed the Louvain method (20.06% mean AMI score). This tends to confirm that modularity-based clustering *à la* Louvain complements the encoding-decoding paradigm of GAE and VGAE, and that learning node embedding spaces from complementary criteria is empirically beneficial. On a more negative note, we nonetheless acknowledge that, on Cora, Citeseer and Pubmed in Table 7, empirical gains of Modularity-Aware models are less visible on graphs *equipped with node features* than on featureless graphs, which we will further discuss in our limitation section.

As our Modularity-Aware models include two main novel components (namely our community-preserving encoders from Section 3.2 and our revised loss from Section 3.3), one might wonder what the contribution of each of these components to the performance gains is. To study this question, we report in Figure 4 the results of an *ablation study*, that consisted in training variant versions of our models leveraging one of these components only[17] (i.e., the encoder but not the loss, or the loss but not the encoder). Figure 4 shows that incorporating any of these two individual components into the model improves community detection. The gain is larger for the *loss* in the Cora example from Figure 4(a), while it is larger for the *encoder* in the Deezer-Album example from Figure 4(b). A simultaneous use of the encoder and of the loss leads to the best results in both examples, which we also confirmed on the other graphs under consideration.

[17] A Modularity-Aware GAE or VGAE model that leverages the novel encoder only (respectively, the novel loss only) corresponds to a particular case of a "complete" Modularity-Aware GAE or VGAE model, where the hyperparameter β (respectively, the hyperparameter λ) is set to 0.

[18] We note that authors of VGAECD-OPT [23] reported larger AMI scores w.r.t. those we managed to obtain from our re-implementation on Cora and Pubmed with features: 54.37% and 35.52%, respectively.

4.2.3. Results for Joint Link Prediction and Community Detection (Task 2)

We now study results for Task 2, the joint link prediction and community detection task described in Section 4.1.2, and performed on incomplete versions of the graph datasets where 15% of edges are randomly masked. Results for this task (i.e., AMI and ARI scores from community detection on incomplete graphs, and AUC and AP scores from link prediction on test sets) are reported in the four rightmost columns of Tables 5, 6 and 7. AMI and ARI scores from this task are also included in the ablation study in Figure 4.

We draw several conclusions from these additional experiments. First of all, we confirm that AMI and ARI scores decrease slightly w.r.t. Task 1, which was expected due to the absence of part of the graph structure during the training phase (e.g., from 46.65% to 42.86% mean AMI, for Linear Modularity-Aware VGAE on Cora in Table 5). Nonetheless, the ranking of the different methods under consideration remains consistent with Task 1. In particular, our Modularity-Aware models still outperform baselines in cases where they were already outperforming in Task 1 (e.g., with a top 43.48% mean AMI for Linear Modularity-Aware GAE on Cora in Table 5, vs 28.41% for the standard Linear GAE and 39.09% for the Louvain method).

Besides these confirmations, the main goal of Task 2 was to address our second research question stated in Section 1: *Do improvements on the community detection task necessarily incur a loss in the link prediction performance or can they be jointly addressed with high accuracy?* Indeed, as GAE and VGAE were originally recognized as effective link prediction methods (see Section 2), improving community detection while deteriorating link prediction might be undesirable, especially in problems requiring effective node embeddings for multitask applications (see the Deezer example from Section 4.1.1). By design, our proposed encoders, losses, and selection procedure specifically aimed to avoid such a deterioration.

Empirical results confirm the ability of Modularity-Aware models to preserve comparable link prediction performances w.r.t standard GAE and VGAE. For instance, in Table 5, our Linear Modularity-Aware GAE reaches mean AUC and AP scores of 87.18% and 88.53%, respectively, which is comparable (or even slightly better in the case of AUC) to Linear Standard GAE (84.46% and 88.42%, respectively). We reach similar results for the three other Modularity-Aware models in Table 5, while scores of several baselines deteriorate by a few points. Overall, all other Modularity-Aware models reported in the complete Table 7 achieve comparable (either better, identical, or only a few points below) AUC and AP scores w.r.t. their GAE or VGAE counterparts.

4.2.4. Limitations and Possible Extensions

As observed in Section 4.2.2, empirical gains of Modularity-Aware models are less pronounced on graphs equipped with node features (although non-null in 2/3 cases). In Table 7, for Cora with features, we "only" report increases in Task 1 AMI scores of +2.63 points w.r.t. the corresponding standard VGAE model. For comparison, in the featureless case, we reported increases in Task 1 AMI scores of +11.53 points. Furthermore, our Modularity-Aware VGAE does not surpass the standard VGAE at all on Pubmed with features. We hypothesize that the incorporation of Louvain-based prior clusters in Modularity-Aware models might be less relevant for these attributed graphs. Indeed, while Louvain only leverages the graph structure for node clustering, node features seem to play a strong role in the identification of ground-truth communities for these graphs.

Nevertheless, we recall that the use of the Louvain method was made without loss of generality. As explained in Section 3.2, our revised message passing operators would remain valid for other methods that alternatively derive a prior clustering signal. Future experiments on such alternatives (e.g., methods processing node features) could therefore improve community detection performances on these three attributed graphs. Overall, the empirical performance of our method directly depends on the quality of the underlying prior clustering method used to compute A_c and A_s, which should therefore be carefully selected.

More broadly, our framework could also straightforwardly incorporate alternative encoders (besides linear and multi-layer GCN encoders), alternative decoders (e.g., decoders replacing inner products by more refined graph reconstruction methods [18, 22]) and alternative losses (for instance, as explained in Section 3.3.1, our modularity-inspired regularizer could be optimized in conjunction with the ELBO loss from VGAECD/VGAECD-OPT [12, 23] involving Gaussian mixtures). One could also replace our k-means step, to cluster vectors z_i, by another method such as k-medoids [97] or spectral clustering [50] (although our preliminary experiments in this direction did not reach significantly better results). Future work considering such alternative architectures for Modularity-Aware GAE and VGAE could definitely lead to the improvement of our models.

Lastly, we also plan to extend Modularity-Aware GAE and VGAE to dynamic graphs. Indeed, while our work considered fixed graph structures, real-world graphs often evolve over time. For instance, on the Deezer service, new albums should regularly appear in the musical catalog. New nodes will therefore appear in the Deezer-Album graph. Capturing such changes, e.g., through dynamic graph embedding methods [2], might permit learning more refined representations and providing effective dynamic community detection.

4.2.5. Towards More Comparisons to Non-GAE/VGAE Methods

Our experiments mainly aim to compare our proposed Modularity-Aware GAE and VGAE models to the Louvain method and to other autoencoders, i.e., 1) the standard GAE and VGAE models, and 2) alternative methods that improve community detection with GAE and VGAE. Giving us a total of 12 baseline models that we compare against. Nonetheless, while our scope is limited to autoencoders, we acknowledge that Modularity-Aware GAE and VGAE could also be seen as multi-task models, as we train them to perform community detection and link prediction simultaneously. As a consequence, in future research, it would be interesting to further study these models through the lens of the multi-task learning framework and thus compare them to other graph-based multi-task learning methods [98, 99].

In addition, there are also interesting parallels between our proposed method and pre-training methods for self-supervised learning (SSL) [100, 101, 102], that might deserve further investigations in future research. There are, however, also several fundamental differences to be pointed out. More specifically, our proposed loss, as is common in the "joint learning" methods in the SSL literature, is also a combination of two loss terms stemming from different learning tasks. However, our method does not involve the distinction between a self-supervised or auxiliary and a downstream or main task; both tasks performed by our method are equally relevant and important. Specifically, with regards to the paradigm of pre-training an encoder on a self-supervised task and then fine-tuning it on a downstream task, our method is trained in a single-stage and does not involve pre-training of any kind. Also, in spirit, our effort in the Modularity-Aware GAE/VGAE is similar to the works of Hu et al. [103] incorporating clustering information

in their cluster-preserving models, the self-supervised graph partition GCN by You et al. [104] and the models predicting degree, local node importance, and local clustering coefficient in Jin et al. [105]. Also, the works addressing label scarcity problems by generating "pseudo-labels" are conceptually similar to our approach since pseudo-labeling often relies on clustering [100, 106, 107]. However, unlike these works, we do not use the community labels in the loss function, but rather optimize the modularity of a clustering obtained from our embedding vectors. In addition, our models do not aim to predict the community membership of a node from its embedding vector, or any other node-level centrality measure such as the degree. Instead, they aim to estimate the likelihood of an edge between two nodes from their two embedding vectors. A more in-depth comparison of our method with this related literature has the potential to yield further improvements of the Modularity-Aware GAE and VGAE that we propose here.

5. Conclusion

In this paper, we introduce a well-performing approach for simultaneous link prediction and community detection compatible with both the GAE and VGAE frameworks. This approach is based on a rigorous diagnosis of the shortcomings of existing approaches to this problem. Our approach takes advantage of two elements: A theoretically grounded variant of message passing operator in the GAE and VGAE encoders, that incorporates prior cluster information, and the addition of a modularity-based loss component to the usual existing loss functions. Both elements were experimentally shown to have an individual impact on the community detection performances. We furthermore introduce a revised hyperparameter selection procedure specifically designed for joint link prediction and community detection. We experimentally demonstrated the effectiveness of the approach on multiple datasets, including common benchmark graphs and large scale real-world ones, both with node features and, crucially featureless graphs: The results are consistently on par or better than the state-of-the-art for both link prediction and community detection. This overcomes the common pitfall of most state-of-the-art methods that usually show high accuracy for either link prediction or community detection but not for both simultaneously. Future research directions include the in-depth analysis of the utilization of the two loss terms and the relationship among them. Specifically, combining a dot-product similarity as a metric for the link prediction term and the Euclidean distance for the modularity-inspired term exhibited strong performance and a study on their behavior appears to be insightful.

Table 7: Summarized results for Task 1 and Task 2 on all graphs. For each graph, for brevity, we only report the **best** Modularity-Inspired model (best on Task 2, among GCN **or** Linear encoder, and GAE **or** VGAE), its standard counterpart, and a comparison to the Louvain baseline as well as the best other baseline (among VGAECD, VGAECD-OPT, ARGA, ARGVA, DVGAE, DeepWalk and node2vec). All node embedding models learn embedding vectors of dimension $d = 16$, with other hyperparameters set as described in Section 4.1.3. Scores are averaged over 100 runs except for the larger SBM and Deezer-Album graphs (10 runs). **Bold** numbers correspond to the best performance for each score. Scores *in italic* are within one standard deviation range from the best score.

Datasets	**Models** (Dimension $d = 16$)	**Task 1: Community Detection on complete graph**		**Task 2: Joint Link Prediction and Community Detection on graph with 15% of edges being masked**			
		AMI (in %)	**ARI (in %)**	**AMI (in %)**	**ARI (in %)**	**AUC (in %)**	**AP (in %)**
Blogs	GCN-based Modularity-Aware VGAE	**73.74 ± 1.32**	**82.78 ± 1.27**	**70.42 ± 1.28**	**79.80 ± 1.12**	**91.67 ± 0.39**	*92.37 ± 0.41*
	GCN-based Standard VGAE	*73.42 ± 0.95*	*82.58 ± 0.93*	66.90 ± 3.32	*77.23 ± 3.89*	*91.64 ± 0.42*	**92.52 ± 0.51**
	Louvain	63.43 ± 0.86	76.66 ± 0.70	57.25 ± 1.67	73.00 ± 1.56	–	–
	Best other baseline:						
	node2vec	72.88 ± 0.87	82.08 ± 0.73	67.64 ± 1.23	77.03 ± 1.85	83.63 ± 0.34	79.60 ± 0.61
Cora	Linear Modularity-Aware GAE	**46.58 ± 0.40**	**39.71 ± 0.41**	**43.48 ± 1.12**	**35.51 ± 1.20**	**87.18 ± 1.05**	**88.53 ± 1.33**
	Linear Standard GAE	35.05 ± 2.55	24.32 ± 2.99	28.41 ± 1.68	19.45 ± 1.75	84.46 ± 1.64	*88.42 ± 1.07*
	Louvain	42.70 ± 0.65	24.01 ± 1.70	39.09 ± 0.73	20.19 ± 1.73	–	–
	Best other baseline:						
	node2vec	41.64 ± 1.25	34.30 ± 1.92	36.25 ± 1.38	29.43 ± 2.21	82.43 ± 1.23	81.60 ± 0.91
Cora with features	Linear Modularity-Aware VGAE	**52.43 ± 1.87**	**44.82 ± 3.12**	**49.48 ± 2.15**	**43.05 ± 3.51**	**93.10 ± 0.88**	**94.06 ± 0.75**
	Linear Standard VGAE	49.98 ± 2.40	*43.15 ± 4.35*	46.90 ± 1.43	38.24 ± 3.56	*93.04 ± 0.80*	*94.04 ± 0.75*
	Louvain	42.70 ± 0.65	24.01 ± 1.70	39.09 ± 0.73	20.19 ± 1.73	–	–
	Best other baseline:						
	VGAECD-OPT	50.32[18] ± 1.95	*43.54 ± 3.23*	47.83 ± 1.64	39.45 ± 3.53	*92.25 ± 1.07*	92.60 ± 0.91
Citeseer	Linear Modularity-Aware VGAE	21.28 ± 1.03	**15.39 ± 1.06**	19.05 ± 1.47	**12.19 ± 1.38**	**80.84 ± 1.64**	**84.21 ± 1.21**
	Linear Standard VGAE	13.83 ± 1.00	8.31 ± 0.89	11.11 ± 1.10	5.87 ± 0.87	78.26 ± 1.55	*82.93 ± 1.39*
	Louvain	**24.72 ± 0.27**	9.21 ± 0.75	**22.71 ± 0.47**	7.70 ± 0.67	–	–
	Best other baseline:						
	node2vec	18.68 ± 1.13	*14.93 ± 1.15*	14.40 ± 1.18	*12.13 ± 1.53*	76.05 ± 2.12	79.46 ± 1.65
Citeseer with features	Linear Modularity-Aware VGAE	**25.11 ± 0.94**	**15.55 ± 0.60**	*22.21 ± 1.24*	**12.59 ± 1.25**	86.54 ± 1.20	88.07 ± 1.22
	Linear Standard VGAE	17.80 ± 1.61	6.01 ± 1.46	17.38 ± 1.43	6.10 ± 1.51	**89.08 ± 1.19**	**91.19 ± 0.98**
	Louvain	24.72 ± 0.27	9.21 ± 0.75	**22.71 ± 0.47**	7.70 ± 0.67	–	–
	Best other baseline:						
	DVGAE	20.09 ± 2.84	12.16 ± 2.74	16.02 ± 3.32	*10.03 ± 4.48*	86.85 ± 1.48	88.43 ± 1.23
Pubmed	Linear Modularity-Aware GAE	**28.54 ± 0.24**	26.36 ± 0.34	**26.38 ± 0.43**	21.30 ± 0.59	**84.39 ± 0.32**	**87.92 ± 0.40**
	Linear Standard GAE	12.61 ± 4.61	6.37 ± 3.86	12.60 ± 4.67	6.21 ± 1.75	82.03 ± 0.32	*87.71 ± 0.24*
	Louvain	20.06 ± 0.27	10.34 ± 0.99	16.71 ± 0.46	8.32 ± 0.79	–	–
	Best other baseline:						
	node2vec	*28.52 ± 1.12*	**30.63 ± 1.14**	23.88 ± 0.54	**25.90 ± 0.65**	81.03 ± 0.30	82.33 ± 0.41
Pubmed with features	Linear Modularity-Aware VGAE	30.09 ± 0.63	**29.11 ± 0.65**	**29.60 ± 0.70**	**28.54 ± 0.74**	*97.10 ± 0.21*	**97.21 ± 0.18**
	Linear Standard VGAE	29.98 ± 0.41	*29.05 ± 0.20*	*29.51 ± 0.52*	*28.50 ± 0.36*	**97.12 ± 0.20**	*97.20 ± 0.17*
	Louvain	20.06 ± 0.27	10.34 ± 0.99	16.71 ± 0.46	8.32 ± 0.79	–	–
	Best other baseline:						
	VGAECD-OPT	**32.47**[18] **± 0.45**	*29.09 ± 0.42*	*29.46 ± 0.52*	*28.43 ± 0.61*	94.27 ± 0.33	94.53 ± 0.36
Cora-Larger	Linear Modularity-Aware VGAE	**48.55 ± 0.18**	**22.21 ± 0.39**	**46.10 ± 0.29**	**20.24 ± 0.41**	**95.76 ± 0.17**	**96.31 ± 0.12**
	Linear Standard VGAE	46.07 ± 0.54	20.01 ± 0.90	43.38 ± 0.37	18.02 ± 0.66	*95.55 ± 0.22*	*96.30 ± 0.18*
	Louvain	44.72 ± 0.50	19.46 ± 0.66	43.41 ± 0.52	19.29 ± 0.68	–	–
	Best other baseline:						
	DVGAE	46.63 ± 0.56	20.72 ± 0.96	43.48 ± 0.61	18.45 ± 0.67	94.97 ± 0.23	95.98 ± 0.21
SBM	Linear Modularity-Aware VGAE	**36.02 ± 0.13**	**8.12 ± 0.06**	**35.85 ± 0.20**	**8.06 ± 0.11**	*82.34 ± 0.38*	*86.76 ± 0.41*
	Linear Standard VGAE	35.01 ± 0.21	7.88 ± 0.15	30.79 ± 0.21	6.50 ± 0.13	80.11 ± 0.35	83.40 ± 0.36
	Louvain	*36.00 ± 0.15*	*8.10 ± 0.15*	*35.84 ± 0.18*	*8.03 ± 0.09*	–	–
	Best other baseline:						
	DVGAE	*35.90 ± 0.18*	*8.07 ± 0.15*	35.53 ± 0.23	*7.95 ± 0.19*	**82.59 ± 0.36**	**87.08 ± 0.40**
Deezer-Album	GCN-Based Modularity-Aware VGAE	**21.64 ± 0.18**	**13.19 ± 0.09**	**19.10 ± 0.21**	**12.00 ± 0.17**	**85.40 ± 0.14**	*86.38 ± 0.15*
	GCN-Based Standard VGAE	15.79 ± 0.32	9.75 ± 0.21	13.98 ± 0.35	8.81 ± 0.32	*85.37 ± 0.12*	**86.41 ± 0.11**
	Louvain	19.81 ± 0.19	12.21 ± 0.09	17.68 ± 0.20	11.02 ± 0.13	–	–
	Best other baseline:						
	node2vec	20.03 ± 0.24	12.20 ± 0.19	18.34 ± 0.29	11.27 ± 0.28	83.51 ± 0.17	84.12 ± 0.15

Appendix A. Proofs of the Propositions in Section 3.2.4

We begin by introducing several theoretical results which we will use in the majority of our proofs. The specific formulations of the results in this section, i.e., Definition 7 and Propositions 8, 9 and 10, are adapted from Lutzeyer [81].

When considering regular graphs, i.e., graphs containing only nodes of equal degree, their different graph representation matrices, such as the adjacency matrix, Laplacian matrices, and the GCN's message passing operator, are related via polynomial matrix transformations. These are now defined.

Definition 7. Horn and Johnson [108, p. 36] define the evaluation of a polynomial $p(x) = c_l x^l + c_{l-1}x^{l-1} + \ldots + c_1 x + c_0$ at a matrix Φ as

$$p(\Phi) = c_l \Phi^l + c_{l-1}\Phi^{l-1} + \ldots + c_1 \Phi + c_0 I.$$

Horn and Johnson [108] further discuss the influence of a polynomial matrix transformation on the matrices' eigenvalues and eigenvectors, which we reproduce below.

Proposition 8. [108] Let $p(\cdot)$ be a given polynomial. If ϕ is an eigenvalue of $\Phi \in \mathbb{R}^{n\times n}$, while u is an associated eigenvector, then $p(\phi)$ is an eigenvalue of the matrix $p(\Phi)$ and u is an eigenvector of $p(\Phi)$ associated with $p(\phi)$.

Since we consider graphs consisting of several connected components in a multitude of our propositions, we now provide a theorem which relates the eigenvalues and eigenvectors of the whole graph to those of its connected components.

Proposition 9. Let $\mathcal{G}$ be a graph with corresponding adjacency matrix A and assume $\mathcal{G}$ to consist of K connected components each with corresponding adjacency matrix A_k for $k \in \{1, \ldots, K\}$. Then, the eigenvalues of $\mathcal{F}_{GCN}(A)$ are equal to the union of the eigenvalues of $\mathcal{F}_{GCN}(A_k)$ over $k \in \{1, \ldots, K\}$. Further, a set of eigenvectors of $\mathcal{F}_{GCN}(A)$ can be constructed from the eigenvector sets of $\mathcal{F}_{GCN}(A_k)$ for $k \in \{1, \ldots, K\}$.

Proof. For a graph $\mathcal{G}$ consisting of several connected components there exists a node ordering such that its corresponding adjacency matrix A is block diagonal. Since, the addition of the identity matrix I_n and the multiplication by a diagonal matrix $(D + I_n)^{-\frac{1}{2}}$ does not affect the block diagonal property of a matrix, the matrix $\mathcal{F}_{GCN}(A)$ is also block diagonal.

Now, the characteristic equation of block diagonal matrices factorizes into polynomials corresponding to the different blocks [109, p. 291]. Therefore, the set of eigenvalues of any block diagonal matrix is equal to the union of the set of eigenvalues of matrices containing only the blocks. Consequently, the eigenvalues of $\mathcal{F}_{GCN}(A)$ are equal to the union of the eigenvalues of $\mathcal{F}_{GCN}(A_k)$ over $k \in \{1, \ldots, K\}$.

Furthermore, any eigenvector of a given connected component, described by $\mathcal{F}_{GCN}(A_k)$, can be modified to be an eigenvector of $\mathcal{F}_{GCN}(A)$ by the insertion of zero values in all entries corresponding to nodes not contained in the connected component described by $\mathcal{F}_{GCN}(A_k)$. □

To allow us to relate the spectra and eigenvectors of the well studied adjacency matrix A to the more novel GCN message passing operator $\mathcal{F}_{GCN}(A)$ we frequently make use of the matrix similarity relationship. The consequences of a matrix similarity relationship between matrices on their eigenvalues and eigenvectors is discussed in Proposition 10.

Proposition 10. [108, pp. 45,60] If two matrices Φ and Ψ are related via a nonsingular matrix S as follows, $\Phi = S^{-1}\Psi S$. Then, Φ and Ψ have the same multiset of eigenvalues. Further, for eigenvector v with corresponding eigenvalue ϕ of Φ gives rise to an eigenvector Sv of Ψ with equal corresponding eigenvalue ϕ.

We can now begin to prove the propositions discussed in Section 3.2.4.

Appendix A.1. Proof of Proposition 4

Proof. Teke and Vaidyanathan [80], among many others, state that the unnormalised Laplacian matrix $L = D - A$ corresponding to a complete graph has eigenvalue 0 with multiplicity 1 and eigenvalue n with multiplicity $n - 1$. Furthermore, the eigenspace corresponding to the eigenvalue n is spanned by a 2-sparse set of orthogonal eigenvectors [80]. In addition, the eigenvector corresponding to the eigenvalue 0 of the unnormalised graph Laplacian describing a connected graph is well known to be the constant eigenvector [50]. Since the complete graph is regular, its degree matrix is a multiple of the identity matrix, i.e., $D = (n-1)I_n$. Therefore, for complete graphs the following relationship holds $\mathcal{F}_{GCN}(A) = I_n - \frac{1}{n}L$. Hence, from Proposition 8 the eigenvectors of $\mathcal{F}_{GCN}(A)$ and L are equal and $\mathcal{F}_{GCN}(A)$ has the eigenvalue 1 with multiplicity 1 and eigenvalue 0 with multiplicity $n - 1$.

Now, since A_c corresponds to a graph composed of several complete graphs, we can invoke Proposition 9 to construct the spectrum and eigenvectors of $\mathcal{F}_{GCN}(A_c)$ from the the spectrum and eigenvectors of $\mathcal{F}_{GCN}(A)$ corresponding to a complete graph, which we just derived. Consequently, $\mathcal{F}_{GCN}(A_c)$ has eigenvalues $\{\{1\}^K, \{0\}^{n-K}\}$ and a set of eigenvectors as described in the statement of Proposition 4.

□

Appendix A.2. Proof of Proposition 5

Proof. Proposition 9 can be used to extend the required result from one connected component of $\mathcal{G}$ to the full graph. Therefore, we consider only one connected component on the graph from now on. Let A' denote the adjacency matrix of this connected component, containing nodes of degree b, and A'_c denote the corresponding complete component of A_c, containing nodes of degree $n' - 1$. Then, the adjacency matrix of our connected component under consideration $A' + \lambda A'_c$ is related to $\mathcal{F}_{GCN}(A' + \lambda A'_c)$ as follows,

$$\mathcal{F}_{GCN}(A' + \lambda A'_c) = \frac{1}{b + \lambda(n'-1) + 1}\left(A' + \lambda A'_c + I\right).$$

Therefore, from Proposition 8 it follows that $A' + \lambda A'_c$ and $\mathcal{F}_{GCN}(A' + \lambda A'_c)$ share eigenvectors. Similarly, the relations $\mathcal{F}_{GCN}(A') = \frac{1}{b+1}(A'+I)$ and $\mathcal{F}_{GCN}(A'_c) = \frac{1}{n'}(A'_c+I)$ together with Proposition 8 allow us to establish that both $\mathcal{F}_{GCN}(A')$ and A' as well as $\mathcal{F}_{GCN}(A'_c)$ and A'_c each have a common set of eigenvectors.

We now make use of a result by Godsil [110, p. 25], which states that the adjacency matrix of a graph commutes with the matrix of all ones, i.e., $A_c + I_n$, if and only if the graph under consideration is regular. Further, a family of matrices is a commuting family if and only if they are simultaneously diagonalizable, i.e., they share a set of eigenvectors [108, p. 52]. Hence, A' and A'_c share a set of eigenvectors. Furthermore, this shared set

of eigenvectors is also a valid set of eigenvectors for $A' + \lambda A'_c$, which, in conjunction with the above polynomial relationships, establishes the needed eigenvector relation.

In addition the eigenvalues of the sum of two simultaneously diagonalizable matrices Φ, Ψ with eigenvalues denoted by ϕ and ψ, respectively, are related [108, p. 54] as follows

$$\mathcal{S}(\Phi + \Psi) = \{\phi_1 + \psi_{s(1)}, \ldots, \phi_n + \psi_{s(n)}\}, \tag{A.1}$$

for some permutation $s(\cdot)$ defined on the set $\{1, \ldots, n\}$. Since A' and $A'_c + I_n$ are simultaneously diagonalizable their eigenvalues follow the relation in Equation (A.1). Now the above polynomial relationships of the GCN message passing operators to the corresponding adjacency matrices gives us the desired eigenvalue result and establish that $g_1(\mu) = \frac{b+1}{b+\lambda(n'-1)+1}(\mu - 1)$ and $g_2(\eta) = \frac{\lambda(n'-1)+1}{b+\lambda(n'-1)+1}(\eta - 1) + 1$.

□

Appendix A.3. Proof of Proposition 6

Proof. Hoory et al. [111, p. 453] state that for s-regular graphs the largest eigenvalue of the corresponding adjacency matrix A' equals s and the corresponding eigenvector is constant. Now the relation $\mathcal{F}_{GCN}(A') = \frac{1}{o+1}(A' + I_n)$ in conjunction with Proposition 8 establish that the largest eigenvalue of $\mathcal{F}_{GCN}(A')$ equals 1 with a corresponding constant eigenvector. This spectrum and eigenvectors can be extended to the matrix $\mathcal{F}_{GCN}(A_s)$ corresponding to a graph of several s-regular connected components using Proposition 9. The comparison of the derived spectrum and eigenvectors to those derived in Proposition 4 completes this proof.

□

References

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

[2] W. L. Hamilton, Graph representation learning, Synthesis Lectures on Artifical Intelligence and Machine Learning 14 (2020) 1–159.

[3] D. Zhang, J. Yin, X. Zhu, C. Zhang, Network representation learning: A survey, IEEE transactions on Big Data (2018).

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, arXiv preprint arXiv:1901.00596 (2019).

[5] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, Journal of the American society for information science and technology 58 (2007) 1019–1031.

[6] A. Kumar, S. S. Singh, K. Singh, B. Biswas, Link prediction techniques, applications, and performance: A survey, Physica A: Statistical Mechanics and its Applications 553 (2020) 124289.

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiments 2008 (2008) P10008.

[8] F. D. Malliaros, M. Vazirgiannis, Clustering and community detection in directed networks: A survey, Physics reports 533 (2013) 95–142.

[9] T. N. Kipf, et al., Deep learning with graph-structured representations, PhD Thesis, University of Amsterdam (2020).

[10] C. Wang, S. Pan, G. Long, X. Zhu, J. Jiang, Mgae: Marginalized graph autoencoder for graph clustering, ACM Conference on Information and Knowledge Management (2017).

[11] T. N. Kipf, M. Welling, Variational graph auto-encoders, NeurIPS Workshop on Bayesian Deep Learning (2016).

[12] J. J. Choong, X. Liu, T. Murata, Learning community structure with variational autoencoder, in: 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 69–78.

[13] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, AAAI Conference on Artificial Intelligence (2014).

[14] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016).

[15] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, International Joint Conference on Artificial Intelligence (2018).

[16] P. V. Tran, Multi-task graph autoencoders, arXiv preprint arXiv:1811.02798 (2018).

[17] G. Salha, R. Hennequin, V. A. Tran, M. Vazirgiannis, A degeneracy framework for scalable graph autoencoders, International Joint Conference on Artificial Intelligence (2019).

[18] G. Salha, S. Limnios, R. Hennequin, V. A. Tran, M. Vazirgiannis, Gravity-inspired graph autoencoders for directed link prediction, ACM International Conference on Information and Knowledge Management (2019).

[19] P.-Y. Huang, R. Frederking, et al., Rwr-gae: Random walk regularization for graph auto encoders, arXiv preprint arXiv:1908.04003 (2019).

[20] A. Grover, A. Zweig, S. Ermon, Graphite: Iterative generative modeling of graphs, International Conference on Machine Learning (2019).

[21] A. Hasanzadeh, E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, X. Qian, Semi-implicit graph variational auto-encoders, Advances in Neural Information Processing Systems (2019).

[22] H. Shi, H. Fan, J. T. Kwok, Effective decoding in graph auto-encoder using triadic closure, AAAI Conference on Artificial Intelligence (2020).

[23] J. J. Choong, X. Liu, T. Murata, Optimizing variational graph autoencoder for community detection with dual optimization, Entropy 22 (2020) 197.

[24] G. Salha, R. Hennequin, J.-B. Remy, M. Moussallam, M. Vazirgiannis, Fastgae: Scalable graph autoencoders with stochastic subgraph decoding, Neural Networks 142 (2021) 1–19.

[25] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, IEEE Transactions on Knowledge and Data Engineering 20 (2007) 172–188.

[26] H. Shiokawa, Y. Fujiwara, M. Onizuka, Fast algorithm for modularity-based graph clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 27, 2013.

[27] G. Salha, R. Hennequin, M. Vazirgiannis, Simple and effective graph autoencoders with one-hop linear models, arXiv preprint arXiv:2001.07614 (2020).

[28] T. Huang, Y. Pei, V. Menkovski, M. Pechenizkiy, On generalization of graph autoencoders with adversarial training, arXiv preprint arXiv:2107.02658 (2021).

[29] J. Li, J. Yu, J. Li, H. Zhang, K. Zhao, Y. Rong, H. Cheng, J. Huang, Dirichlet graph variational autoencoder, Advances in Neural Information Processing Systems 33 (2020).
[30] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, International Conference on Learning Representations (2017).
[31] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, International Conference on Learning Representations (2014).
[32] J. Chen, T. Ma, C. Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, International Conference on Learning Representations (2018).
[33] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019).
[34] W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in Neural Information Processing Systems (2017).
[35] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Advances in Neural Information Processing Systems (2016).
[36] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[37] J. Park, M. Lee, H. J. Chang, K. Lee, J. Y. Choi, Symmetric graph convolutional autoencoder for unsupervised graph representation learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6519–6528.
[38] J. Li, T. Yu, D.-C. Juan, A. Gopalan, H. Cheng, A. Tomkins, Graph autoencoders with deconvolutional networks, arXiv preprint arXiv:2012.11898 (2020).
[39] Q. Liu, M. Allamanis, M. Brockschmidt, A. Gaunt, Constrained graph variational autoencoders for molecule design, Advances in Neural Information Processing Systems (2018).
[40] M. Simonovsky, N. Komodakis, Graphvae: Towards generation of small graphs using variational autoencoders, International Conference on Artificial Neural Networks (2018).
[41] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
[42] M. Fey, J. E. Lenssen, Fast graph representation learning with PyTorch Geometric, ICLR Workshop on Representation Learning on Graphs and Manifolds (2019).
[43] D. P. Kingma, M. Welling, Auto-encoding variational bayes, International Conference on Learning Representations (2014).
[44] W. Jin, R. Barzilay, T. Jaakkola, Junction tree variational autoencoder for molecular graph generation, International Conference on Machine Learning (2018).
[45] S. Kullback, R. A. Leibler, On information and sufficiency, The Annals of Mathematical Statistics 22-1 (1951) 79–86.
[46] C. Doersch, Tutorial on variational autoencoders, arXiv preprint arXiv:1606.05908 (2016).
[47] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (2008) 93–93.
[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[49] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2014).
[50] U. Von Luxburg, A tutorial on spectral clustering, Statistics and computing 17 (2007) 395–416.
[51] Y. Hao, X. Cao, Y. Fang, X. Xie, S. Wang, Inductive link prediction for nodes having only attribute information, International Joint Conference on Artificial Intelligence (2020).
[52] V. Rennard, G. Nikolentzos, M. Vazirgiannis, Graph auto-encoders for learning edge representations, in: International Conference on Complex Networks and Their Applications, 2020, pp. 117–129.
[53] R. v. d. Berg, T. N. Kipf, M. Welling, Graph convolutional matrix completion, KDD Deep Learning Day (2018).
[54] X. Wu, Q. Cheng, Deepened graph auto-encoders help stabilize and enhance link prediction, arXiv preprint arXiv:2103.11414 (2021).
[55] G. Salha-Galvan, R. Hennequin, B. Chapus, V.-A. Tran, M. Vazirgiannis, Cold start similar artists ranking with gravity-inspired graph autoencoders, 15th ACM Conference on Recommender Systems (2021).
[56] Y. Kaloga, P. Borgnat, S. P. Chepuri, P. Abry, A. Habrard, Multiview variational graph autoencoders for canonical correlation analysis, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 5320–5324.
[57] T. Ma, J. Chen, C. Xiao, Constrained generation of semantically valid graphs via regularizing

variational autoencoders, Advances in Neural Information Processing Systems (2018).
[58] S. Fortunato, Community detection in graphs, Physics Reports 486 (2010) 75–174.
[59] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, E. Cambria, Learning community embedding with community detection and node embedding on graphs, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 377–386.
[60] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, L. Lin, A unified framework for community detection and network representation learning, IEEE Transactions on Knowledge and Data Engineering 31 (2018) 1051–1065.
[61] F.-Y. Sun, M. Qu, J. Hoffmann, C.-W. Huang, J. Tang, vgraph: A generative model for joint community detection and node representation learning, Advances in Neural Information Processing Systems 32 (2019).
[62] D. He, Y. Song, D. Jin, Z. Feng, B. Zhang, Z. Yu, W. Zhang, Community-centric graph convolutional network for unsupervised community detection, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 3515–3521.
[63] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, Oakland, CA, USA, 1967, pp. 281–297.
[64] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, International Conference on Machine Learning (2019).
[65] K. Greff, S. Van Steenkiste, J. Schmidhuber, Neural expectation maximization, Advances in Neural Information Processing Systems (2017).
[66] G. Dasoulas, J. F. Lutzeyer, M. Vazirgiannis, Learning parametrised graph shift operators, in: International Conference on Learning Representations, 2021.
[67] V. Waradpande, D. Kudenko, M. Khosla, Graph-based state representation for deep reinforcement learning, arXiv preprint arXiv:2004.13965 (2020).
[68] J. Shin, K. Kim, D. Park, S. Kim, J. Kang, Bipartite link prediction by intra-class connection based triadic closure, IEEE Access 8 (2020) 140194–140204.
[69] G. Salha, R. Hennequin, M. Vazirgiannis, Keep it Simple: Graph Autoencoders Without Graph Convolutional Networks, NeurIPS 2019 Workshop on Graph Representation Learning (2019).
[70] M. E. J. Newman, Modularity and community structure in networks, Proceedings of the National Academy of Sciences 103 (2006) 8577–8582.
[71] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Processing Magazine (2013) 83 – 98.
[72] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: 2nd International Conference on Learning Representations (ICLR), 2014.
[73] R. Levie, F. Monti, X. Bresson, M. M. Bronstein, Cayleynets: Graph convolutional neural networks with complex rational spectral filters, IEEE Transactions on Signal Processing 67 (2019) 97–109.
[74] F. R. K. Chung, Spectral graph theory, 92, Providence, R.I.: American Mathematical Society, 1997.
[75] D. Spielman, Spectral graph theory, in: Combinatorial scientific computing, CRC Press, 2012.
[76] A. Sandryhaila, J. M. Moura, Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure, IEEE Signal Processing Magazine 31 (2014) 80–90.
[77] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proceedings of the IEEE 106 (2018) 808–828.
[78] F. Gama, A. Ribeiro, J. Bruna, Stability of graph neural networks to relative perturbations, in: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 9070–9074.
[79] M. Balcilar, G. Renton, P. Héroux, B. Gaüzère, S. Adam, P. Honeine, Analyzing the expressive power of graph neural networks in a spectral perspective, in: International Conference on Learning Representations, 2021.
[80] O. Teke, P. P. Vaidyanathan, Uncertainty principles and sparse eigenvectors of graphs, IEEE Transactions on Signal Processing 65 (2017) 5406–5420.
[81] J. Lutzeyer, Network representation matrices and their eigenproperties: A comparative study, PhD Thesis: Imperial College London, 2020.
[82] H. Weyl, Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung), Mathematische Annalen 71 (1912) 441–479.
[83] L. Y. Kolotilina, The strengthened versions of the additive and multiplicative weyl inequalities,

Journal of Mathematical Sciences 127 (2005) 1976–1987.
[84] J. F. Lutzeyer, A. T. Walden, Extending the davis-kahan theorem for comparing eigenvectors of two symmetric matrices i: Theory, arXiv preprint arXiv:1908.03462 (2019).
[85] J. Friedman, A proof of alon's second eigenvalue conjecture, in: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, 2003, pp. 720–724.
[86] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Thirty-first AAAI conference on artificial intelligence, 2017.
[87] X. Liu, C. Zhuang, T. Murata, K.-S. Kim, N. Kertkeidkachorn, How much topological structure is preserved by graph embeddings?, Computer Science and Information Systems 16 (2019) 597–614.
[88] L. Yang, X. Cao, D. He, C. Wang, X. Wang, W. Zhang, Modularity based community detection with deep learning., in: International Joint Conference on Artificial Intelligence, 2016.
[89] I. Lobov, S. Ivanov, Unsupervised community detection with modularity-based attention model, arXiv preprint arXiv:1905.10350 (2019).
[90] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (2014) 1929–1958.
[91] E. Abbe, Community detection and stochastic block models: recent developments, The Journal of Machine Learning Research 18 (2017) 6446–6531.
[92] D. Arthur, S. Vassilvitskii, K-means++: The advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, 2007, p. 1027–1035.
[93] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (2015).
[94] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), 2016, pp. 265–283.
[95] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016).
[96] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of Machine Learning Research 9(11) (2008).
[97] H.-S. Park, C.-H. Jun, A simple and fast algorithm for k-medoids clustering, Expert Systems with Applications 36 (2009) 3336–3341.
[98] Z. Wu, M. Zhan, H. Zhang, Q. Luo, K. Tang, Mtgcn: A multi-task approach for node classification and link prediction in graph data, Information Processing & Management 59 (2022) 102902.
[99] S. Wang, Q. Wang, M. Gong, Multi-task learning based network embedding, Frontiers in Neuroscience (2020) 1387.
[100] Y. Xie, Z. Xu, J. Zhang, Z. Wang, S. Ji, Self-supervised learning of graph neural networks: A unified review, arXiv preprint arXiv:2102.10757 (2021).
[101] Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, P. S. Yu, Graph self-supervised learning: A survey, arXiv preprint arXiv:2103.00111 (2021).
[102] L. Wu, H. Lin, C. Tan, Z. Gao, S. Z. Li, Self-supervised learning on graphs: Contrastive, generative, or predictive, IEEE Transactions on Knowledge and Data Engineering (2021).
[103] Z. Hu, C. Fan, T. Chen, K.-W. Chang, Y. Sun, Pre-training graph neural networks for generic structural feature extraction, arXiv preprint arXiv:1905.13728 (2019).
[104] Y. You, T. Chen, Z. Wang, Y. Shen, When does self-supervision help graph convolutional networks?, in: international conference on machine learning, PMLR, 2020, pp. 10871–10880.
[105] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, J. Tang, Self-supervised learning on graphs: Deep insights and new direction, arXiv preprint arXiv:2006.10141 (2020).
[106] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI conference on artificial intelligence, 2018.
[107] K. Sun, Z. Lin, Z. Zhu, Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 5892–5899.
[108] R. A. Horn, C. R. Johnson, Matrix Analysis, Cambridge, UK: Cambridge University Press, 1985.
[109] D. S. Bernstein, Matrix mathematics: Theory, facts, and formulas (second edition), Princeton, NJ:Princeton University Press, 2009.
[110] C. Godsil, Algebraic combinatorics, volume 6, CRC Press, 1993.
[111] S. Hoory, N. Linial, A. Wigderson, Expander graphs and their applications, Bulletin of the American Mathematical Society (2006) 439 – 561.