New Frontiers in Graph Autoencoders: Joint Community Detection and Link Prediction

Guillaume Salha-Galvan*
Deezer Research
Paris, France

Johannes F. Lutzeyer
LIX, École Polytechnique, IP Paris
Palaiseau, France

George Dasoulas
DBMI, Harvard University
Cambridge, MA, USA

Romain Hennequin
Deezer Research
Paris, France

Michalis Vazirgiannis
LIX, École Polytechnique, IP Paris
Palaiseau, France

Abstract

Graph autoencoders (GAE) and variational graph autoencoders (VGAE) emerged as powerful methods for link prediction (LP). Their performances are less impressive on community detection (CD), where they are often outperformed by simpler alternatives such as the Louvain method. It is still unclear to what extent one can improve CD 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 LP in a multi-task setting. In this workshop paper, summarizing results from our journal publication [44], we show that jointly addressing these two tasks with high accuracy is possible. For this purpose, we introduce a community-preserving message passing scheme, doping our GAE and VGAE encoders by considering both the initial graph and Louvain-based prior communities when computing embedding spaces. Inspired by modularity-based clustering, we further propose novel training and optimization strategies specifically designed for joint LP and CD. We demonstrate the empirical effectiveness of our approach, referred to as Modularity-Aware GAE and VGAE, on various real-world graphs.

1 Introduction

Extracting relevant information from nodes of a graph is essential to tackle a wide range of machine learning problems [9, 13, 14, 57]. This includes *link prediction* (LP) [26, 29], which consists in inferring the presence of new or unobserved edges between node pairs, and *community detection* (CD) [4, 9], which consists in clustering nodes into similar groups, according to a chosen similarity metric. To address such problems, significant efforts have recently been devoted to the development of *node embedding* methods [13, 14, 23]. These methods aim to learn vectorial representations of nodes in an *embedding space* where node positions should reflect and summarize the initial graph structure. They assess the probability of a new edge between two nodes, or their likelihood of belonging to the same community, by evaluating the proximity of these nodes in the embedding space [7, 24, 53].

In particular, *graph autoencoders* (GAE) and *variational graph autoencoders* (VGAE) [24, 49, 53, 54] recently emerged as two powerful families of node embedding methods. Both methods rely on an *encoding-decoding* strategy that consists in *encoding* nodes into an embedding space from which *decoding*, i.e., reconstructing the original graph, should ideally be possible. Originally mainly designed for LP (at least in their modern formulation leveraging *graph neural networks* (GNN) [24]), the effectiveness of GAE and VGAE models and their extensions on this task has been experimentally

*Corresponding author at: research@deezer.com.

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

confirmed [12, 16, 18, 36, 43, 45, 50]. On the other hand, several studies [7, 8, 40, 41] have pointed out their limitations on CD. These studies emphasized that GAEs and VGAEs are often outperformed by simpler CD alternatives, such as the popular Louvain method [4]. The question of how to improve CD with GAEs and VGAEs remains incompletely addressed, especially in the absence of node features. Moreover, it is still unclear to which extent one can improve CD with these models without simultaneously deteriorating LP, and jointly address these two problems. These questions are highly relevant in practice, as learning node embedding spaces suitable for multi-task settings leads to consistent inference between tasks and saves costs in real-world applications.

This paper[2] presents several contributions pushing the frontiers of GAEs and VGAEs, and showing that jointly addressing CD and LP with high accuracy is possible with these models. After reviewing key concepts in Section 2, we explain why GAEs and VGAEs underperform on CD in Section 3. We simultaneously introduce *Modularity-Aware GAE and VGAE*, our solution leveraging *modularity-based clustering* concepts [4, 5, 46] to improve CD while preserving the ability to identify missing edges in LP. We report an in-depth evaluation of our method in Section 4, and conclude in Section 5.

2 Preliminaries

We consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes and m edges. We denote its $n \times n$ adjacency matrix by A. Each node $i \in \mathcal{V}$ is equipped with a feature vector $x_i \in \mathbb{R}^f$. We denote the $n \times f$ matrix having x_i vectors as rows by X. For a featureless graph, we set $X = I_n$, the identity matrix.

GAE and VGAE. The term GAE refers to a family of unsupervised two-component models learning node embedding spaces in the absence of node labels [23, 24, 49, 54]. The first component is the *encoder*, a parameterized function processing A and X, and mapping each node $i \in \mathcal{V}$ to an *embedding vector* $z_i \in \mathbb{R}^d$, with $d \ll n$. In practice, a GNN [13, 25, 57] often acts as the encoder, i.e., $Z = \text{GNN}(A, X)$, with Z the $n \times d$ matrix having z_i vectors as rows. The second component is the *decoder*, estimating an adjacency matrix $\hat{A}$ from embedding vectors: $\hat{A} = \text{Decoder}(Z)$. Decoders can be neural networks, or simpler functions, e.g., based on inner products between z_i vectors [24, 28, 37, 54]. When training a GAE, one wishes to learn z_i vectors from which reconstructing $\mathcal{G}$ should be possible. Intuitively, this would indicate that the embedding space preserves some important information about $\mathcal{G}$. For this purpose, model weights are trained via gradient descent minimization [10] of a *reconstruction loss*, usually a cross entropy [24], evaluating the similarity between $\hat{A}$ and A.

Introduced as probabilistic extensions of GAEs, VGAE models associate z_i vectors with distributions. Notably, in the seminal VGAE from Kipf and Welling [24], each vector $z_i \sim \mathcal{N}(\mu_i, \Sigma_i)$. Their model incorporates *two* GNN encoders processing both A and X: one of them learns mean vectors $\mu_i \in \mathbb{R}^d$, and the other learns variance matrices $\Sigma_i \in \mathbb{R}^{d \times d}$, for all $i \in \mathcal{V}$. Moreover, instead of a reconstruction loss, they optimize the variational *evidence lower bound* (ELBO) of the model's likelihood [22], using gradient ascent. Besides constituting promising generative models [20, 30, 47], variants of VGAEs also turned out to be effective alternatives to GAEs in some LP and CD tasks [8, 16, 24, 40, 42, 43].

Evaluation. Over the past years, LP[3] has become the most prominent way to evaluate the quality of embedding vectors learned from a GAE or VGAE [12, 15, 16, 18, 19, 36, 43, 45, 50]. Previous work widely confirmed the effectiveness of GAEs and VGAEs on this task. Their performances are less impressive on CD[3], another important graph problem with numerous applications [6, 17, 34, 48, 51]. In the presence of node embedding representations z_i, CD boils down to the common problem of clustering n vectors, e.g., via a k-means [33] in the embedding space. Nonetheless, concurring work [7, 8, 40, 41] recently pointed out the limitation of this approach for GAEs and VGAEs, and its lower performance w.r.t. simpler CD alternatives, such as the popular Louvain method learning communities by iteratively maximizing the density-based *modularity* value in the graph [4].

While recent studies aimed to address the underwhelming performance of GAEs and VGAEs on CD, they still suffer from limitations that motivate our work. Firstly, several studies [7, 8, 27] considered

[2] This workshop paper summarizes results from our journal article *"Modularity-Aware Graph Autoencoders for Joint Community Detection and Link Prediction"* accepted for publication in Elsevier's Neural Networks journal in 2022 [44]. The purpose of our submission to GLFrontiers was to present this work to a live audience.

[3] We provide more formal presentations of the LP and CD problems under consideration in Appendix A.

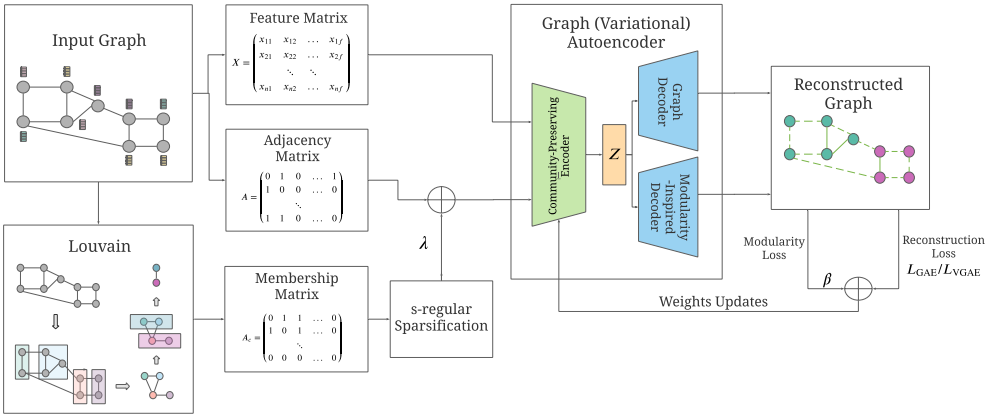


Figure 1: Overview of our proposed Modularity-Aware GAE/VGAE. Firstly, input graph data A and X are combined with the s-regular sparsified prior community membership matrix A_s, derived through iterative modularity maximization via the Louvain algorithm, as described in the first paragraph of Section 3. Then, they are processed by our revised community-based 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 the second paragraph of Section 3. Furthermore, other hyperparameters from this model are tuned via the method described in the third paragraph of Section 3 and designed for joint LP and CD.

clustering-oriented probabilistic priors for VGAEs (such as Gaussian mixtures in VGAECD [7] and VGAECD-OPT [8]), that cannot be transposed to the deterministic GAE setting. Secondly, a closer look at these models reveals that their empirical gains mostly stem from the addition of node features. They offer little advantage when features are absent (see Table 4). Other studies did not consider featureless graphs at all [18, 19, 36, 37, 45]. This motivates the need to investigate GAE/VGAE-based CD on featureless graphs. Thirdly, previous studies did not try to preserve good performances on LP [7, 8, 27, 53]. It is still uncertain whether one can jointly address LP and CD with accuracy in multi-task settings, which, as argued in the Introduction, is highly relevant in practice. In conclusion, the question of improving CD with GAEs and VGAEs remains incompletely addressed.

3 Modularity-Aware GAE and VGAE for Joint LP and CD

To address these limitations, we introduce our Modularity-Aware GAE/VGAE, illustrated in Figure 1.

Community-Based Encoders. Firstly, we argue that most GAEs/VGAEs leverage encoders that do not specifically aim to capture graph communities. This includes graph convolutional networks (GCN) [25], which remain the most popular encoders in practice [12, 16, 18, 19, 36, 45], and encoders identifying clusters from features rather than the graph [8, 53]. Modularity-Aware GAE and VGAE overcome this issue by incorporating a *community-based encoder*.

Specifically, we first obtain a partition of the node set using the Louvain method [4] and store it in an $n \times n$ membership matrix A_c, defined as $(A_c)_{ij} = 1$ if nodes $i \neq j$ are in the same community, and 0 otherwise. Then, when learning embedding vectors, we leverage this partition as a *prior signal*, from which the encoder should benefit, but also have the ability to deviate. Formally, we replace the $Z = \text{GNN}(A, X)$ component[4] by: $Z = \text{GNN}(A + \lambda A_s, X)$, where $\lambda \in \mathbb{R}^+$ and $s \in \mathbb{N}^+$ are hyperparameters, and where A_s is a s-regular sparsified[5] version of A_c. This change alters the GNN *message passing scheme*. Nodes will now aggregate information from their neighbors *and* some nodes of their prior community (λ balances the importance of these two information sources). Therefore, nodes from the same prior community will tend to have more similar embedding vectors than with a standard GAE or VGAE.

[4]For clarity of exposition we discuss the deterministic GAE framework. Our modifications equally apply to the VGAE framework, for which Z has to be replaced by Gaussian parameters (see Section 2).

[5]In A_s, nodes are only connected to s fixed and randomly selected neighbors from their community. This sparsification permits speeding up GNN message passing operations in practice [25].

Besides its simplicity and good performance on CD [40], our justification for using Louvain as a prior is threefold. Firstly, it automatically selects the relevant number of prior communities to consider. Secondly, it runs in $O(n \log n)$ time [4] and, therefore, scales to graphs with millions of nodes. Thirdly, it optimizes a *modularity* criterion that complements the encoding-decoding paradigm. We will show in Section 4 that learning representations from complementary criteria is beneficial. Nonetheless, our framework remains valid for any alternative method providing prior communities.

Modularity-Inspired Losses. Previous models were also *trained* in a fashion that, by design, favors LP over CD. The cross entropy and ELBO losses involve the reconstruction of *node pairs* from the embedding space [24]. However, a good reconstruction of *local* pairwise connections does not necessarily imply a good reconstruction of the *global* community structure [31, 55]. Consequently, in Modularity-Aware GAE (respectively, VGAE), we minimize (resp., maximize), using gradient descent (resp., gradient ascent), an alternative function that subtracts (resp., adds) the following *global regularizer* to the cross entropy (resp., ELBO) term: $\frac{\beta}{2m} \sum_{i,j=1}^{n} [A_{ij} - \frac{d_i d_j}{2m}] e^{-\gamma \|z_i - z_j\|_2^2}$, with d_i the degree [13] of node $i \in \mathcal{V}$ and two hyperparameters $\beta \in \mathbb{R}^+$ and $\gamma \in \mathbb{R}^+$.

A soft and differentiable version of the *modularity* [35] (independent of any ground truth community), this regularizer aims to push closer vectors z_i of densely connected parts of the graph, and, therefore, to permit a k-means-based detection of communities with higher density. Several studies out of the GAE/VGAE scope emphasized the effectiveness of comparable approaches for learning community-preserving representations [32, 55, 56]. On the other hand, the remaining presence of the local cross entropy (resp., ELBO) in our optimized loss aims to preserve good performances on LP. β balances the relative importance of the global regularizer. γ regulates the magnitude of $\|z_i - z_j\|_2^2$ in the exponential term, which tends to 1 when z_i and z_j get closer, and to 0 when they move apart.

Hyperparameter Selection. GAEs and VGAEs include several important hyperparameters such as dropout and learning rates [24] (our models also introduce λ, s, β, and γ). In previous studies, their selection procedure was sometimes solely based on LP validation sets [40, 41]. However, optimal values for CD might differ from those for LP, partly explaining the low performance on CD. In this paper, we consider an alternative hyperparameter selection procedure. As detailed in Appendix A, the hyperparameters selected for our models are chosen by maximizing the average of: (1) an *Area under the ROC Curve (AUC) score* computed on an LP validation set, and (2) the *modularity* score computed from the communities extracted from final vectors z_i, via a k-means. We expect this dual criterion to identify hyperparameters that will be jointly relevant for LP and CD in a multi-task setting.

4 Experimental Evaluation

We now report results from an in-depth experimental evaluation of our method. Our code is available on GitHub: `https://github.com/GuillaumeSalhaGalvan/modularity_aware_gae`.

Setting. For evaluation, we consider a "pure" CD problem, as well as a multi-task LP/CD problem, on seven graphs of various origins and sizes (from 1124 to 2.5 million nodes). For both problems and all graphs, we compare our approach to 12 baselines, including the Louvain method [4], standard GAE/VGAE models [24] with varying encoders, and existing extensions of GAEs/VGAEs for CD. For brevity, we report technical details on tasks, datasets, models, and hyperparameters in Appendix A.

Results on CD. CD results from Table 3 confirm the discussed limitations of standard GAE/VGAE, which Louvain outperforms on 5 of 7 featureless graphs (e.g., +7.45 Adjusted Mutual Information (AMI) points for Louvain on Pubmed). On the contrary, our Modularity-Aware GAE/VGAE almost always surpass the Louvain method, *and* the use of a standard GAE/VGAE (e.g., with a top 21.64% AMI on the largest Album graph). Interestingly, combining Louvain and a GAE/VGAE into our Modularity-Aware models is beneficial even when the GAE/VGAE initially outperforms Louvain (e.g., for Cora-Large). This confirms that modularity-based clustering *à la* Louvain complements the encoding-decoding paradigm, and that leveraging complementary criteria is empirically beneficial. We also compare favorably to other baselines in most experiments (e.g., +2.11 AMI points w.r.t. VGAECD-OPT [8] on Cora with features), with or without the addition of node features. Figure 2 provides a visualization of node embedding representations learned by our models.

Results on Multi-Task CD/LP. We now assess whether improving CD implies deteriorating the effectiveness on LP. The last columns of Table 3 confirm the ability of Modularity-Aware GAE/VGAE to preserve good performances on LP (we achieve comparable scores w.r.t standard GAE/VGAE on all graphs). While performances on CD decrease slightly w.r.t. pure CD (an expected result, as some edges are masked during training for the purpose of LP), we continue to outperform baselines in most experiments. This demonstrates the effectiveness of our approach at jointly addressing CD and LP.

Discussion on Model Components. For most models, using a linear encoder [42] gives competitive LP/CD results w.r.t. a 2-layer GCN [24]. Also, VGAE models often outperform their GAE counterparts, even though scores are relatively close. Our proposed hyperparameter selection procedure had a noticeable impact on the choices of λ, β, γ, and s, as well as on the required number of training iterations, which we illustrate in Figure 3. In such cases, optimal values for joint LP and CD differ from those for LP only. Lastly, one might wonder whether our performance gains mainly come from our novel encoder or our regularized loss. Figure 4 reports an *ablation study*, consisting in training variant versions of Modularity-Aware VGAEs with one component only (i.e., the novel encoder but not the regularized loss, or vice versa). We show that incorporating any of these two individual contributions into the VGAE improves CD, and that their simultaneous use leads to the best results.

5 Conclusion

In this paper, we introduced a well-performing approach for joint CD and LP with GAEs and VGAEs. We demonstrated its effectiveness through in-depth experimental validation. Our work paves the way for various future research, including replacing Louvain with other prior methods, using our regularizer in conjunction with other reconstruction losses (e.g., ELBO variants computed from Gaussian mixtures [7, 8]), and extending our approach to dynamic graphs. The journal version[2] of this work [44] includes several additional extensions as well as results, omitted here for brevity. This includes further comparisons to non-GAE/VGAE methods, a spectral analysis of our message passing operator, and discussions on how this research helps the music streaming service Deezer address real-world multi-task LP and CD problems for music recommendation purposes.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Dystems Design and Implementation*. 265–283.

[2] Emmanuel Abbe. 2017. Community Detection and Stochastic Block Models: Recent Developments. *The Journal of Machine Learning Research* 18, 1 (2017), 6446–6531.

[3] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. 1027–1035.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiments* 2008, 10 (2008), P10008.

[5] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. 2007. On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering* 20, 2 (2007), 172–188.

[6] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. Learning Community Embedding with Community Detection and Node Embedding on Graphs. In *2017 ACM on Conference on Information and Knowledge Management*.

[7] Jun Jin Choong, Xin Liu, and Tsuyoshi Murata. 2018. Learning Community Structure with Variational Autoencoder. In *2018 IEEE International Conference on Data Mining*.

[8] Jun Jin Choong, Xin Liu, and Tsuyoshi Murata. 2020. Optimizing Variational Graph Autoencoder for Community Detection with Dual Optimization. *Entropy* 22, 2 (2020), 197.

[9] Santo Fortunato. 2010. Community Detection in Graphs. *Physics Reports* 486, 3-5 (2010), 75–174.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[12] Aditya Grover, Aaron Zweig, and Stefano Ermon. 2019. Graphite: Iterative Generative Modeling of Graphs. *International Conference on Machine Learning* (2019).

[13] William L Hamilton. 2020. Graph Representation Learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14, 3 (2020), 1–159.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin* (2017).

[15] Yu Hao, Xin Cao, Yixiang Fang, Xike Xie, and Sibo Wang. 2020. Inductive Link Prediction for Nodes Having Only Attribute Information. *International Joint Conference on Artificial Intelligence* (2020).

[16] Arman Hasanzadeh, Ehsan Hajiramezanali, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Semi-Implicit Graph Variational Auto-Encoders. *Advances in Neural Information Processing Systems* (2019).

[17] Dongxiao He, Yue Song, Di Jin, Zhiyong Feng, Binbin Zhang, Zhizhi Yu, and Weixiong Zhang. 2021. Community-Centric Graph Convolutional Network for Unsupervised Community Detection. In *International Joint Conference on Artificial Intelligence*. 3515–3521.

[18] Po-Yao Huang, Robert Frederking, et al. 2019. RWR-GAE: Random Walk Regularization for Graph Auto Encoders. *arXiv preprint arXiv:1908.04003* (2019).

[19] Tianjin Huang, Yulong Pei, Vlado Menkovski, and Mykola Pechenizkiy. 2021. On Generalization of Graph Autoencoders with Adversarial Training. *arXiv preprint arXiv:2107.02658* (2021).

[20] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. *International Conference on Machine Learning* (2018).

[21] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (2015).

[22] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *International Conference on Learning Representations* (2014).

[23] Thomas N Kipf et al. 2020. Deep Learning with Graph-Structured Representations. *PhD Thesis, University of Amsterdam* (2020).

[24] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NeurIPS Workshop on Bayesian Deep Learning* (2016).

[25] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations* (2017).

[26] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. 2020. Link Prediction Techniques, Applications, and Performance: A Survey. *Physica A: Statistical Mechanics and its Applications* 553 (2020), 124289.

[27] Jia Li, Jianwei Yu, Jiajin Li, Honglei Zhang, Kangfei Zhao, Yu Rong, Hong Cheng, and Junzhou Huang. 2020. Dirichlet Graph Variational Autoencoder. *Advances in Neural Information Processing Systems* 33 (2020).

[28] Jia Li, Tomas Yu, Da-Cheng Juan, Arjun Gopalan, Hong Cheng, and Andrew Tomkins. 2020. Graph Autoencoders with Deconvolutional Networks. *arXiv preprint arXiv:2012.11898* (2020).

[29] David Liben-Nowell and Jon Kleinberg. 2007. The Link-Prediction Problem for Social Networks. *Journal of the American Society for Inf. Sci. and Technology* 58, 7 (2007), 1019–1031.

[30] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. Constrained Graph Variational Autoencoders for Molecule Design. *Advances in Neural Information Processing Systems* (2018).

[31] Xin Liu, Chenyi Zhuang, Tsuyoshi Murata, Kyoung-Sook Kim, and Natthawut Kertkeidkachorn. 2019. How Much Topological Structure is Preserved by Graph Embeddings? *Computer Science and Information Systems* 16, 2 (2019), 597–614.

[32] Ivan Lobov and Sergey Ivanov. 2019. Unsupervised Community Detection with Modularity-based Attention Model. *arXiv preprint arXiv:1905.10350* (2019).

[33] James MacQueen et al. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. Oakland, CA, USA, 281–297.

[34] Fragkiskos D Malliaros and Michalis Vazirgiannis. 2013. Clustering and Community Detection in Directed Networks: A Survey. *Physics reports* 533, 4 (2013), 95–142.

[35] M. E. J. Newman. 2006. Modularity and Community Structure in Networks. *Proceedings of the National Academy of Sciences* 103, 23 (2006), 8577–8582.

[36] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. *International Joint Conference on Artificial Intelligence* (2018).

[37] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6519–6528.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[39] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014).

[40] Guillaume Salha, Romain Hennequin, Jean-Baptiste Remy, Manuel Moussallam, and Michalis Vazirgiannis. 2021. FastGAE: Scalable Graph Autoencoders with Stochastic Subgraph Decoding. *Neural Networks* 142 (2021), 1–19.

[41] Guillaume Salha, Romain Hennequin, Viet Anh Tran, and Michalis Vazirgiannis. 2019. A Degeneracy Framework for Scalable Graph Autoencoders. *International Joint Conference on Artificial Intelligence* (2019).

[42] Guillaume Salha, Romain Hennequin, and Michalis Vazirgiannis. 2020. Simple and Effective Graph Autoencoders with One-Hop Linear Models. *arXiv preprint arXiv:2001.07614* (2020).

[43] Guillaume Salha, Stratis Limnios, Romain Hennequin, Viet Anh Tran, and Michalis Vazirgiannis. 2019. Gravity-Inspired Graph Autoencoders for Directed Link Prediction. *ACM International Conference on Information and Knowledge Management* (2019).

[44] Guillaume Salha-Galvan, Johannes F Lutzeyer, George Dasoulas, Romain Hennequin, and Michalis Vazirgiannis. 2022. Modularity-Aware Graph Autoencoders for Joint Community Detection and Link Prediction. *Neural Networks* 153 (2022), 474–495.

[45] Han Shi, Haozheng Fan, and James T Kwok. 2020. Effective Decoding in Graph Auto-Encoder using Triadic Closure. *AAAI Conference on Artificial Intelligence* (2020).

[46] Hiroaki Shiokawa, Yasuhiro Fujiwara, and Makoto Onizuka. 2013. Fast Algorithm for Modularity-Based Graph Clustering. In *AAAI Conference on Artificial Intelligence*, Vol. 27.

[47] Martin Simonovsky and Nikos Komodakis. 2018. GraphVAE: Towards Generation of Small Graphs using Variational Autoencoders. *International Conference on Artificial Neural Networks* (2018).

[48] Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. 2019. vgraph: A Generative Model for Joint Community Detection and Node Representation Learning. *Advances in Neural Information Processing Systems* 32 (2019).

[49] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning Deep Representations for Graph Clustering. *AAAI Conference on Artificial Intelligence* (2014).

[50] Phi Vu Tran. 2018. Multi-Task Graph Autoencoders. *arXiv preprint arXiv:1811.02798* (2018).

[51] Cunchao Tu, Xiangkai Zeng, Hao Wang, Zhengyan Zhang, Zhiyuan Liu, Maosong Sun, Bo Zhang, and Leyu Lin. 2018. A Unified Framework for Community Detection and Network Representation Learning. *IEEE Transactions on Knowledge and Data Engineering* 31, 6 (2018), 1051–1065.

[52] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9(11), 11 (2008).

[53] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. MGAE: Marginalized Graph Autoencoder for Graph Clustering. *ACM International Conference on Information and Knowledge Management* (2017).

[54] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[55] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *Thirty-first AAAI conference on artificial intelligence*.

[56] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. 2016. Modularity Based Community Detection with Deep Learning. In *International Joint Conference on Artificial Intelligence*.

[57] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network Representation Learning: A Survey. *IEEE Transactions on Big Data* (2018).

Appendix

This appendix provides details on our experimental setting in Appendix A, and complementary tables and figures from our experiments in Appendix B.

A Experimental Setting

Datasets. We consider seven graphs of various origins, characteristics, and sizes. Firstly, we study the *Cora* ($n = 2708$, $m = 5429$), *Citeseer* ($n = 3327$, $m = 4732$) and *Pubmed* ($n = 19717$, $m = 44338$) citation networks [25], *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, acting as the communities to be detected. These graphs are commonly used to evaluate GAEs and VGAEs. We, therefore, see value in studying them as well, especially in their *featureless* version where previous GAE and VGAE extensions fall short on CD.

In addition, we consider a larger version of Cora, referred to as *Cora-Large* ($n = 23166$, $m = 91500$) [42]. Nodes are documents clustered in 70 topic-related communities. Additionally, we consider the *Blogs web graph* ($n = 1224$, $m = 19025$) [42]. Nodes correspond to webpages of political blogs connected through hyperlinks, and clustered in two communities corresponding to politically left-leaning or right-leaning blogs. Thirdly, we examine the *SBM* graph ($n = 100000$, $m = 1498844$), generated from a *stochastic block model*, i.e., a generative model for community-based random graphs [2]. 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 CD methods.

Lastly, we consider *Album* ($n = 2503985$, $m = 25039155$) a private graph provided by the music streaming service Deezer. Nodes are *music albums* available on this service, connected through an undirected edge when they are regularly *co-listened* to by users. The service is jointly interested in (1) predicting new connections in the graph, corresponding to new albums pairs that users would enjoy listening to together; 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). In such an application, learning album representations that would *jointly* enable effective LP and CD would therefore be desirable. For evaluation, 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 node communities.

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

- **Task 1:** We first examine a "pure" *CD* 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 a k-means (with k-means++ initialization [3]) in the final embedding space of each model to cluster the vectors z_i, with k matching the known number of communities; except for some baseline methods that explicitly incorporate another strategy to partition nodes. We compare the obtained partitions to the ground truth using the *Adjusted Mutual Information (AMI)* and *Adjusted Rand Index (ARI)* scores[6].

- **Task 2:** We also study a *joint LP and CD* task. In such a *multi-task* 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) 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., LP, from the embedding space, using the *Area Under the ROC Curve (AUC)* and *Average Precision (AP)* scores[6]. Jointly, we evaluate the CD performance obtained from such incomplete graphs, using the same methodology and scores as in Task 1.

[6] Scores are computed via scikit-learn, using formulas provided in the sklearn.metrics documentation [38].

Table 1: Complete list of optimal hyperparameters of Modularity-Aware GAE and VGAE models.

Dataset	Learning rate	Number of iterations	Dropout rate	Use of FastGAE [40] (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
Album	0.005	600	0.0	Yes (10 000)	0.25	0.25	1	5

In the case of Task 2, we expect AMI and ARI scores to decrease w.r.t. Task 1, as models will only observe *incomplete* versions of the graphs when learning embedding spaces. With Task 2, we aim to assess whether improving CD inevitably leads to deteriorating performances on LP.

Models: Details on the Hyperparameter Selection Procedure. For these two tasks and seven graphs, we compare the performances of our proposed Modularity-Aware GAE and VGAE to standard GAE and VGAE and to several other baselines. All models described below will verify $d = 16$ (the journal version of this work also discusses results obtained with $d \in \{32, 64\}$, which lead to similar conclusions as $d = 16$). We choose other hyperparameters using the *selection procedure* mentioned in Section 3, and further described in the next paragraph.

Foremost, as CD is an unsupervised task, we cannot rely on train/validation/test splits as for the supervised LP classification task[7]. Consistently with our other contributions, we rather rely on the *modularity* [35], an unsupervised density-based criterion computed independently of ground truth communities. Precisely, we select hyperparameters that maximize the average of:

- the AUC obtained for LP on the validation set of Task 2;
- the modularity: $Q = \frac{1}{2m}\sum_{i,j=1}^{n}[A_{ij} - \frac{d_i d_j}{2m}]\delta(i,j)$, computed from the communities extracted by running a k-means on the final vectors z_i, learned from the train graph of Task 2. In this equation, $\delta(i,j) = 1$ if nodes i and j belong to the same community and 0 otherwise.

We expect this dual criterion to identify hyperparameters jointly relevant to LP and CD.

Models: Modularity-Aware GAE and VGAE. We trained two versions of our Modularity-Aware GAE and VGAE: one with the *linear encoder* proposed by Salha et al. [42], and one with the *2-layer GCN encoder* used by Kipf and Welling [24]. The latter encoder includes a 32-dimensional hidden layer. As most GAE/VGAE models, we use a simple inner product decoder: $\hat{A}_{ij} = \sigma(z_i^T z_j)$.

During training, we used the Adam optimizer [21], without dropout (but we tested models with dropout values in $\{0, 0.1, 0.2\}$ in our grid search optimization). For each graph, we considered 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 1. We adopted the same optimal hyperparameters for GAE *and* VGAE variants. Lastly, as the exact loss computation was computationally infeasible for our two largest graphs, SBM and Album, their corresponding models were trained by using the FastGAE method [40], approximating losses by reconstructing degree-based sampled subgraphs of $n = 10000$ nodes (a different one at each training iteration).

We used Tensorflow [1], 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[8].

[7] Ground truth communities are *unavailable* during training. They will only be revealed for model evaluation, to compare the agreement of the node partition inferred by each model to the ground truth partition.

[8] On our machines, running times of the Modularity-Aware GAE and VGAE 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 $s = 5$ takes 25 minutes on a single GPU (without FastGAE).

Models: Standard GAE and VGAE. We examine two variants of the standard GAE and VGAE: one with 2-layer GCN encoders with a 32-dimensional hidden layer (which is equal to the GAE and VGAE from Kipf and Welling [24]) and one with a linear encoder (which is equal to the linear GAE and VGAE from Salha et al. [42]). We note that these models 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, LP is performed from inner product decoding, and CD via a k-means on vectors z_i. We selected similar learning rates and numbers of iterations to the values reported in Table 1.

Models: Other Baselines. We also report experiments on VGAECD [7], a *VGAE for CD* model that replaces Gaussian priors by learnable *Gaussian mixtures*. Such a change permits recovering communities from node embedding spaces without relying on an additional k-means step. We also tested VGAECD-OPT, an improved version of VGAECD by the same authors [8]. Specifically, VGAECD-OPT replaces GCN encoders with linear models. It also adopts a different optimization procedure based on neural expectation-maximization, which guarantees that communities do not collapse during training and experimentally leads to better performances [8]. We set similar hyper-parameters to the above other GAE/VGAE-based models. In all models, the number of Gaussian mixtures matches the ground truth number of communities in each graph.

Besides, we also report experiments on the *Dirichlet Graph Variational Autoencoder* (DGVAE) [27], another extension of VGAE which uses Dirichlet distributions as priors on latent vectors, acting as indicators of community membership. We set similar learning rates and layer dimensions to the above GAE/VGAE-based models. In the case of DGVAE, we use 2-layer GCN encoders for consistency with other models in our experiments. We nonetheless acknowledge that the authors also proposed another encoder, 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 examine the *Adversartially-Regularized (Variational) Graph Autoencoder* (ARGA and ARVGA) models [36], that incorporate an adversarial regularization scheme to GAE and VGAE, with similar hyperparameters as previous models. ARGA and ARVGA emerged as some of the most cited GAE/VGAE extensions and, while they were not specifically introduced for CD, Pan et al. [36] reported empirical gains on this task w.r.t. standard GAE/VGAE, on graphs with node features.

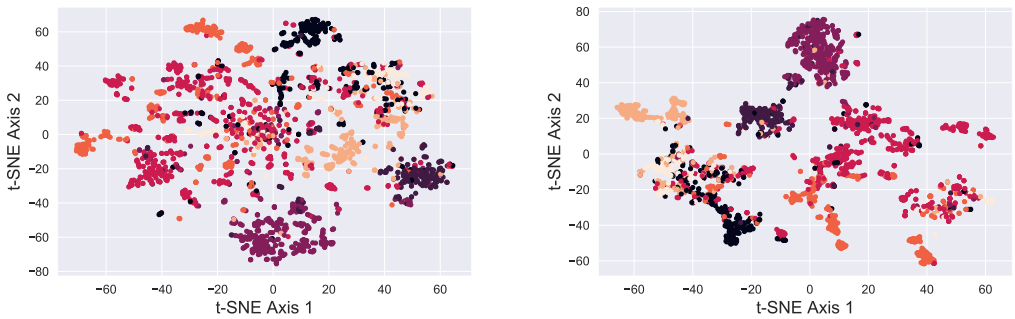
For completeness, we add three baselines not utilizing the autoencoder paradigm. We report results obtained from the popular node embedding methods *node2vec* [11] and *DeepWalk* [39], training models from 10 random walks with length 80 per node, a window size of 5 and on a single epoch. For node2vec, we further set $p = q = 1$. We use a similar strategy as GAEs/VGAEs (k-means/inner products) for CD and LP from embedding spaces. Lastly, we also compare to the *Louvain* method [4] for CD. We see value in comparing to a direct use of Louvain, as this method is directly leveraged in our Modularity-Aware GAE/VGAE as a pre-processing step for the computation of A_c and A_s.

B Figures and Tables

We now provide complementary tables and figures from our experiments. Table 2 details complete results for the Cora dataset and Table 3 reports more summarized results for several more graphs.

Table 2: Results for Task 1 and Task 2 on the featureless Cora graph, using Modularity-Aware GAE/VGAE with Linear and GCN encoders, their standard GAE/VGAE counterparts, and other baselines. All node embedding models learn vectors of dimension $d = 16$. Scores are averaged over 100 runs. LP results are reported from test sets. **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

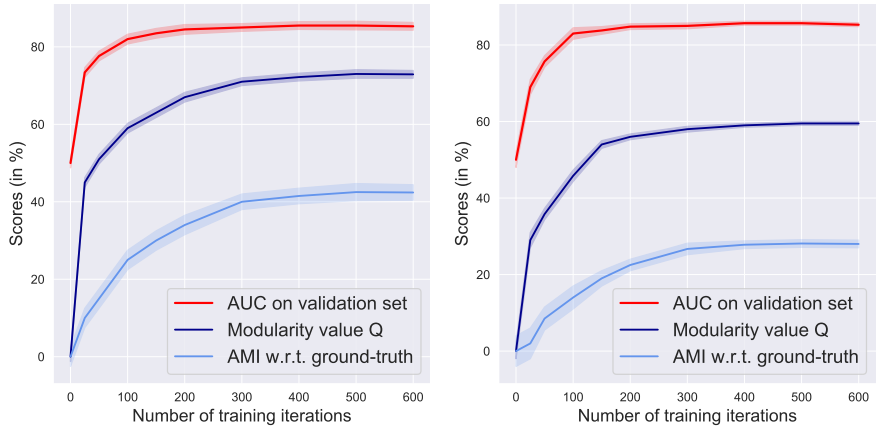


(a) Linear Standard VGAE (b) Linear Modularity-Aware VGAE

Figure 2: 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 [52]. Colors denote ground truth communities, that were not available during training. Although CD is not perfect (both methods return AMI scores $< 50\%$ in Table 2), node embedding representations from (b) provide a more visible separation of these communities. Specifically, in Table 2, using Linear Modularity-Aware VGAE for CD leads to an increase of 9 AMI points (Task 1) to 10 AMI points (Task 2) for CD w.r.t. Linear Standard VGAE, while preserving comparable performances in LP (Task 2).

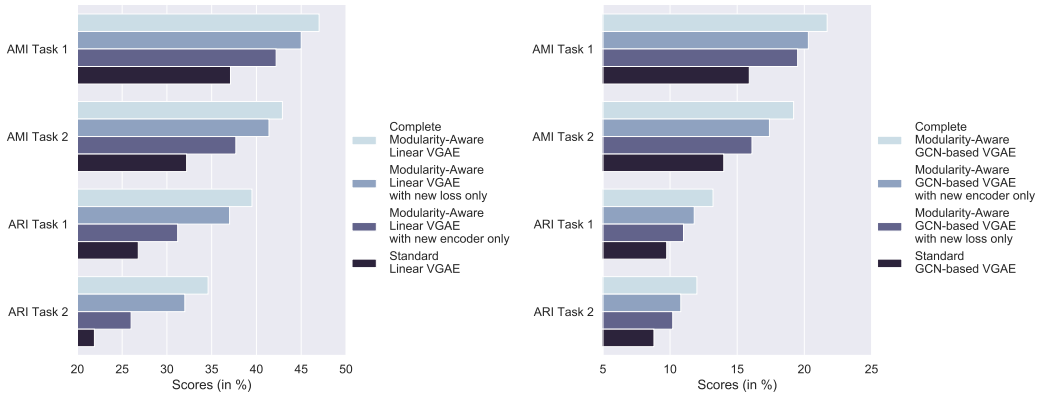
Table 3: Summarized results for Task 1 and Task 2 on all graphs. For each graph, for brevity, we only report the **best** Modularity-Inspired model (best on Task 2, among GCN **or** Linear encoder, and GAE **or** VGAE), its standard counterpart, and a comparison to the Louvain baseline as well as the best other baseline (among VGAECD, VGAECD-OPT, ARGA, ARGVA, DVGAE, DeepWalk, and node2vec). All node embedding models learn vectors of dimension $d = 16$. Scores are averaged over 100 runs except for SBM and Album (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 ± 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 ± 0.45**	*29.09 ± 0.42*	*29.46 ± 0.52*	*28.43 ± 0.61*	94.27 ± 0.33	94.53 ± 0.36
Cora-Large	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**
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



(a) Cora (b) Pubmed

Figure 3: 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 LP scores on validation sets (red) w.r.t. the number of training iterations in gradient ascent. By looking at the red curves only, one might choose to stop training models after 200 iterations as in [24], as AUC 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 unavailable at training time.



(a) Cora (b) Album

Figure 4: Comparison of two "complete" Modularity-Aware VGAE, trained on (a) featureless Cora and (b) Album with variants of these models only leveraging our new *encoder* or regularized *loss* from Section 3. We observe that incorporating any of these two components improves CD on these graphs w.r.t. Standard VGAE. Moreover, using both components *simultaneously* leads to the best results.

Table 4: Normalized mutual information scores (in %) for CD on Cora and Pubmed, *with* and *without* node features. *Results are directly taken from the evaluation of Choong et al.* [8]. 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