Project Report for the Probabilistic Graphical Models and Deep Generative Networks Course

Master Mathématiques, Vision, Apprentissage
ENS Paris-Saclay

Name: Younes BOUKACEM, Tristan MARTIN

Research Project Title: Learning deep representations for graph clustering

1 Introduction

Graph clustering is a problem where the goal is to uncover hidden node clusters or "communities" within a graph. This problem underlies many real-world tasks such as image segmentation (Shi & Malik, 2000), community detection (White & Smyth, 2005), and integrated circuits design (Chan et al., 1993). Classical spectral clustering (Ng et al., 2001) provides a principled approach to graph clustering, but its reliance on the eigenvalue decomposition of the graph similarity matrix can limit not only its scalability (complexity varies from super-quadratic to $O(N^3)$ with N being the number of nodes), but also its flexibility—namely, the ability to add constraints to the problem formulation.

To overcome these limitations, Tian et al. (2014) proposed GraphEncoder, a deep learning model that employs stacked sparse autoencoders to learn nonlinear embeddings of the normalized similarity matrix. The learned representations preserve graph structure while enabling efficient clustering via K-Means. The method is theoretically motivated by the equivalence between spectral clustering and low-rank reconstruction objectives, establishing a link between spectral methods and autoencoders.

The objective of this project is twofold. First, we reproduce and critically evaluate the GraphEncoder model by providing a rigorous analysis of its theoretical foundations and an empirical benchmarking of its performance against spectral clustering and K-Means on a selected set of real-world and synthetic graphs. Second, we extend the original framework by introducing a Denoising GraphEncoder, motivated by the manifold hypothesis (Chapelle, 2006), to improve the performance of GraphEncoder and reduce sensitivity to hyperparameter choices. Through extensive experimentation, we aim to clarify the strengths, limitations, and practical trade-offs of deep autoencoder-based approaches to graph clustering.

2 Summary, Benchmarking, and Limitations of Tian et al. (2014)

2.1 The Problem: Graph Clustering

Tian et al. (2014) propose a method to solve the graph clustering problem. In this problem, we are given as input a weighted, undirected graph whose nodes are grouped into disjoint, hidden clusters that we seek to discover. Formally, we consider the input to be a graph $G = (V, E)$ where $V = \{v_1, ..., v_n\}$ denotes the set of nodes, and $E = \{e_{ij} \in \mathbb{R} \mid i, j \in [1..n]\}$ the set of weighted edges connecting the unordered pairs of nodes $\{v_i, v_j\} \in V \times V$ (i.e. $e_{ij} = e_{ji} \forall i, j$). Also, we suppose that the nodes of V are organized according to a latent partition $P = \{V_1, ..., V_K\}$ of V (i.e. $\cup_{c=1}^{K} V_c = V$ and $V_c \cap V_l = \phi,\ \forall c \neq l \in [1..K]$). The goal is to recover P.

2.2 Motivating the Use of Autoencoders for Graph Clustering

In the scenario considered by Tian et al. (2014), no prior information is provided on P, leading to an unsupervised problem which requires to introduce reasonable assumptions. To do so, they propose to build on the approach of normalized cut ($NCut$) minimization, where we assume that P contains K clusters and minimizes a predefined objective as given by Equation 1:

$$P = \underset{\{V_1,...V_K\}}{\operatorname{argmin}} \ NCut = \sum_{c=1}^{K} \frac{link(V_c, \bar{V_c})}{volume(V_c)}\ , \quad \text{where: } \begin{cases} \bar{V_c} := \bar{V_c} \cap V_c = \phi, \bar{V_c} \cup V_c = V \\ link(V_c, \bar{V_c}) = \sum\limits_{i \in V_c, j \in \bar{V_c}} e_{ij} \\ volume(V_c) = \sum\limits_{i \in V_c} \sum\limits_{j \in V} e_{ij} \end{cases} \tag{1}$$

Loosely speaking, minimizing the $NCut$ objective ensures that, overall, the clusters of P have less outgoing connections than intra-cluster connections. Now, if we define the two matrices D and $S \in \mathbb{R}^{n \times n}$ to be $D = diag(d_1, ..., d_n)$ where $d_i = \sum_{j \in V} e_{ij}$ (that is, D is the diagonal matrix of nodes degrees d_i), and $S = (e_{ij})_{i,j \in [1..n]}$ (that is, S is the similarity

matrix of the graph G), it can be shown (see subsection A.1) that by leveraging some relaxation, solving the NP-Hard combinatorial optimization defined by Equation 1 amounts to finding the best rank-K matrix approximation of $D^{-1}S$ in terms of the Frobenius norm, i.e. finding the matrix $\mathcal{O}$ defined by:

$$\mathcal{O} = \underset{X \in \mathbb{R}^{n \times n},\ rank(X)=K}{\operatorname{argmin}} \|D^{-1}S - X\|_F \tag{2}$$

in which case, by the *Eckart-Young-Mirsky* theorem, $\mathcal{O} = Y_K \Lambda_K Y_K^T$ where Λ_K is the diagonal matrix of the K largest eigen-values of $D^{-1}S$, and $Y_k \in \mathbb{R}^{n \times K}$ are their associated eigen-vectors. Each row in Y_K is then used as an embedding of its corresponding node in $D^{-1}S$, and the clusters are obtained by applying the K-means algorithm on those embeddings. This relaxation approach, known as SPECTRAL-CLUSTERING (SC), motivates Tian et al. (2014) to apply the autoencoder architecture to $D^{-1}S$—which they consider as a data matrix of n samples of dimension n— arguing that the auto-encoder accomplishes a similar low-rank matrix approximation task as in Equation 2. Indeed, this can be visualized by considering a linear auto-encoder with latent dimension $h < n$. By design, such a model attempts to produce a matrix $\mathcal{O}$ given by Equation 3:

$$\mathcal{O} = \underset{X \in \mathbb{R}^{n \times d},\ X = D^{-1}SW_\nabla W_\Delta}{\operatorname{argmin}} \mathcal{L}(D^{-1}S - X) \tag{3}$$

where $W_\nabla \in \mathbb{R}^{n \times h}, W_\nabla \in \mathbb{R}^{h \times n}$ are the encoder and decoder weights respectively, and $\mathcal{L}$ is the network's loss function computing the disparity between $D^{-1}S$ and X, akin to a Frobenius norm. By construction, $rank(\mathcal{O}) = rank(D^{-1}SW_\nabla W_\Delta) \leq min(n, h) = h$, effectively performing a low-rank approximation of $D^{-1}S$. However, autoencoders, while theoretically similar to spectral clustering, are far more flexible in adding constraints such as sparse latent representations, which can improve storage efficiency and clustering accuracy. Indeed, autoencoders can easily achieve this using sparse regularization terms in their loss function and backpropagation. Also, stacking multiple layers of sparse autoencoders can achieve additional performance from deep structures.

2.3 The Solution: GraphEncoder

Based on this analysis, Tian et al. (2014) propose to train a stacked sparse auto-encoder to reconstruct $D^{-1}S$, and apply the K-means algorithm on the latent space representation obtained at the bottle-neck. Specifically, their solution, named GRAPHENCODER (GE), consists of choosing a number T of one-deep auto-encoders equipped with a non-linear activation function (such as a sigmoid or tanh activation) at both encoder and decoder ends. After ordering the auto-encoders in decreasing size of their latent dimension from AE_1 to AE_T, we begin by training AE_1 on reconstructing $D^{-1}S$ through backpropagation on the loss given by Equation 4:

$$\mathcal{L}_{AE} = \underbrace{\sum_{i=1}^{n} \|x_i - AE_1(x_i)\|_2}_{(*)} + \underbrace{\beta \mathrm{KL}(\rho \| \hat{\rho})}_{(**)}, \tag{4}$$

where $(*)$ is the reconstruction penalty, and $(**)$ is the sparsity penalty (detailed in subsubsection A.2.2). The latent space representation $H_1 = Encoder_1(D^{-1}S)$ is then given as input to the second auto-encoder AE_2 which is trained similarly and so on. The process terminates by retrieving the latent space representation H_T associated with AE_T, and generating the clusters by applying K-means on H_T.

2.4 Benchmarking GraphEncoder

While Tian et al. (2014) present the results of several experiments, we could not find any released code to assess their method. Consequently, we decide to implement a PyTorch (Paszke et al., 2019) replica of GraphEncoder and test it on a selected set of real and synthetic benchmarks of reduced size, abiding by our computational limits.

Specifically, we benchmark GRAPHENCODER on 5 real world benchmarks: KARATE($|V| = 34, K = 2$)(Zachary, 1977), FOOTBALL($|V| = 115, K = 12$)(Girvan & Newman, 2002), POLBOOKS($|V| = 105, K = 3$)(Adamic & Glance, 2005), EMAIL($|V| = 1005, K = 42$)(Leskovec et al., 2007), and WINE($|V| = 178, K = 3$)(Dua & Graff, 2019). For the synthetic benchmarks, we use the LFR algorithm (Lancichinetti et al., 2008) to generate artificial graphs with ($|V| = 250, K = 3$): the LFR algorithm generates random networks with realistic community structure and known ground truth. A key parameter of LFR is the mixing parameter μ which controls how many edges a node has outside its cluster. As μ increases, clusters become less separated and the benchmark becomes progressively harder. Given the substantial number of hyper-parameters present in GRAPHENCODER (such as $T, h_1, ...h_T, \beta, \rho$ etc.), we perform automated hyper-parameter tuning using Optuna (Akiba et al., 2019), optimizing for the NCUT instead of the NMI so as not to bias the hyper-parameter selection via ground-truth knowledge, and thus provide a fair comparison with spectral clustering and direct K-Means. We relegate a more detailed description of the experimental setup to Appendix B, and present here the obtained results in Table 1 and Figure 1.

Method	Karate	Football	Polbooks	Email	Wine
Normalized Cut (NCut). Lower is better.					
K-means (**ncut-km**)	0.1338	3.6736	0.3169	11.1010	1.8980
Spectral Clustering (**ncut-sc**)	0.1338	3.6736	0.2751	20.6253	1.8958
Graph Encoder (**ncut-ge**)	0.1338	3.7032	0.2751	10.2216	1.8980
Normalized Mutual Information (NMI). Higher is better.					
K-means (**nmi-km**)	0.8371	0.9241	0.5165	0.4086	0.6351
Spectral Clustering (**nmi-sc**)	0.8371	0.9241	0.5744	0.4871	0.7126
Graph Encoder (**nmi-ge**)	0.8371	0.9268	0.5744	0.4143	0.6351
Graph Encoder (**best-nmi-ge**)	0.8371	0.9325	0.5744	0.6659	0.6351

Table 1: Benchmarking results on real-world graphs comparing K-means(km), Spectral Clustering(sc), and Graph Encoder(ge).

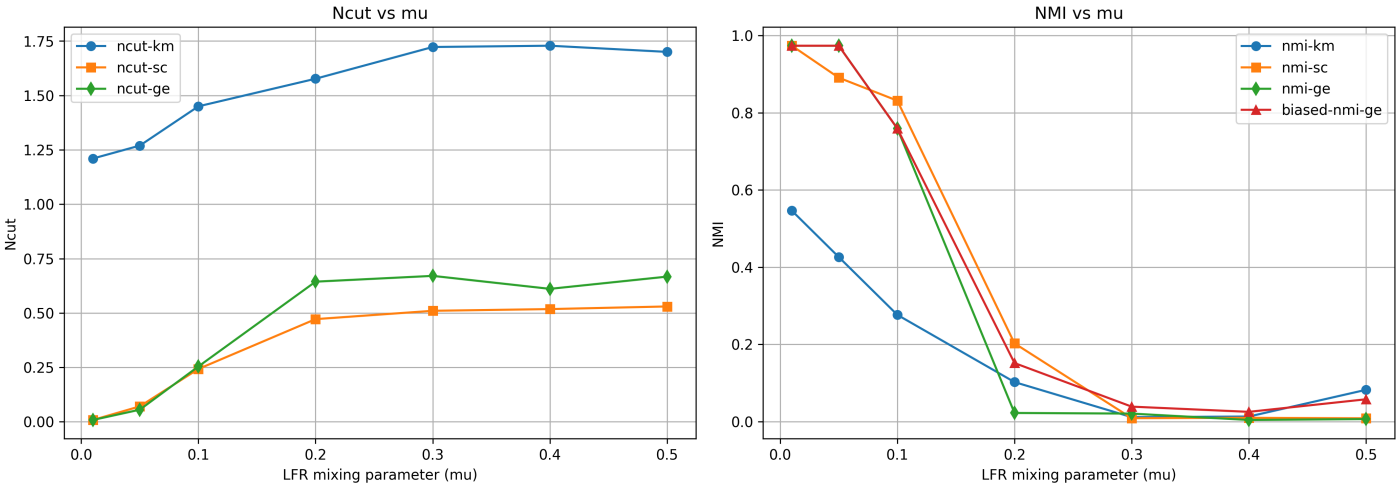

Figure 1: Ncut and NMI results against synthetic datasets generated by LFR (mu).

1. Description of the reported metrics: For each benchmark, we report the following metrics:

1. **ncut-km, nmi-km:** NCut and NMI obtained by applying K-Means directly to the graph's normalized similarity matrix $D^{-1}S$. To account for K-Means variability, we run the algorithm $N = 100$ times with different seeds and retain the clustering with the lowest inertia to compute the metrics.
2. **ncut-sc, nmi-sc:** NCut and NMI obtained by applying spectral clustering to S. As above, the K-Means step is repeated N times.
3. **ncut-ge, nmi-ge:** NCut and NMI obtained by applying GraphEncoder using the best hyperparameter configuration found when optimizing for NCut. Metrics are computed after running K-Means N times on the learned latent representations.
4. **best-nmi-ge:** The best NMI achieved across all tested GraphEncoder hyperparameter configurations that were obtained during tuning for NCut. While not used for comparison with K-Means or spectral clustering (as it is based on knowing the ground truth), we find this metric is useful for the concluding discussion in subsection 2.5.

2. Analysis of results on real-world benchmarks: From Table 1, we observe that GraphEncoder generally outperforms direct K-Means on both NCut and NMI. The sole exception is NCut on Football, where K-Means exceeds GraphEncoder by approximately 0.8%. When compared to spectral clustering, GraphEncoder achieves competitive but not consistently superior results. For NCut, GraphEncoder matches spectral clustering on Karate and Pollbooks, underperforms on Football and Wine, and surpasses it by roughly 50% on Email. A similar pattern emerges for NMI, though not across the same datasets: GraphEncoder again matches spectral clustering on Karate and Pollbooks, outperforms it on Football by about 0.3%, and underperforms on Email and Wine by approximately 17.6% and 12.2%, respectively.

3. Analysis of results on LFR benchmarks: From Figure 1, we see that all methods degrade as the mixing parameter μ increases, reflecting the increasing difficulty of LFR benchmarks. Overall, Spectral-Clustering and GraphEncoder achieve the strongest and closely competing performance, while K-Means lags significantly. This beavior is particularly pronounced for low μ (i.e., < 0.1), where SC and GE exhibit very low NCut values and near-perfect NMI. As μ grows, all methods deteriorate rapidly in both metrics, with SC and GE showing comparable rates of performance decline.

2.5 Conclusions regarding strengths and limitations of GraphEncoder

Consistent with the findings of Tian et al. (2014), our results show that GraphEncoder reliably improves upon direct K-Means. However, unlike their reported outcomes, our experiments indicate that GraphEncoder does not consistently outperform spectral clustering by a significant margin on either real-world or synthetic graphs; rather, the two methods perform comparably. Lacking access to their original implementation, we hypothesize that their stronger results may stem from more extensive hyperparameter tuning. Supporting this hypothesis, several of our tested benchmarks (e.g., Football and Email) show that when considering the **best-nmi-ge** metric, the GraphEncoder model achieves notably higher NMI than spectral clustering (see Table 1; Football: $0.9325 > 0.9268$, Email: $0.6659 > 0.4871$). This suggests that additional tuning could lead to improved performance, but also highlights a practical limitation: obtaining competitive results may require substantial computational resources. As an additional note, GraphEncoder does not estimate the number of clusters automatically, which may limit its applicability in scenarios where this quantity is unknown; though this aspect is not a primary concern in the original work, as the method explicitly assumes the number of clusters to be known a priori.

3 Improvement: Robust Feature Learning via Denoising Autoencoders

3.1 Theoretical Formulation: From Reconstruction to Denoising

As discussed in subsection 2.5, the primary limitation of the GraphEncoder proposed by Tian et al. (2014) is its sensitivity to hyperparameter choices, which necessitates extensive tuning. To address this issue, we consider the approach of improving the geometric quality of the learned node embeddings so that they better reflect the latent community structure of the graph, thus potentially reducing sensitivity to specific hyperparameter choices and improving model performance.

To this end, we note that the standard GraphEncoder is based on a simple reconstruction autoencoder. However, as noted in the literature (Goodfellow et al., 2016; Rifai et al., 2011; Vincent et al., 2010), minimizing the reconstruction error can be insufficient to guarantee the extraction of expressive features, as the autoencoder may simply learn a trivial identity mapping—copying the input to the output—without capturing the underlying geometric structure of the data within its latent space. To mitigate this issue, we propose replacing the sparse stacked autoencoder of Tian et al. (2014) with a sparse stacked **denoising** autoencoder (Vincent et al., 2010).

Formally, let x_i denote the i-th row of the normalized similarity matrix $D^{-1}S$. We introduce a stochastic corruption process $q_{\mathcal{D}}(\tilde{x}_i|x_i)$ that transforms the original input x_i into a corrupted version $\tilde{x}_i$. The objective function is modified such that the autoencoder AE (at any level of the stack) takes the corrupted $\tilde{x}_i$ as input but is optimized to reconstruct the *clean*, uncorrupted input x_i. The loss function of this denoinsing autoencoder DAE is thus defined by Equation 5:

$$\mathcal{L}_{DAE} = \sum_{i=1}^{n} \|x_i - AE(\tilde{x}_i)\|_2 + \beta \mathrm{KL}(\rho \| \hat{\rho}) \tag{5}$$

For the specific case of graph clustering, we employ a **Masking Noise** as the corruption process. In this setup, a fraction ν of the elements in x_i (chosen at random for each example) is forced to 0, while the remaining elements are kept the same. We name this approach Denoising GraphEncoder (DGE).

3.2 Geometric Interpretation and Relevance to Graph Clustering

We argue that the introduction of the denoising criterion, beyond being a mere regulazization technique, may help to introduce the required inductive biases during the learning process of the node embeddings in order for them to include community aware geometric properties.

Specifically, the geometric interpretation of this denoising technique relies on the manifold hypothesis (Chapelle, 2006; Fefferman et al., 2016), which states that natural high-dimensional data concentrates close to a non-linear low-dimensional manifold. Under this hypothesis:

- The corrupted points $\tilde{x_i}$ are likely to fall outside and farther from the manifold than their uncorrupted counterparts x_i.
- By minimizing the reconstruction error against the clean input, the DAE learns a stochastic operator $p(X|\tilde{X})$ that projects these low-probability corrupted points back onto nearby high-probability points on the manifold.
- Consequently, the embedding representation $Z = Encoder(X)$ serves as a coordinate system for points on the manifold, capturing the main variations in the data structure.

Application to graph clustering Based on these assumptions, the denoising approach would be theoretically better suited for graph clustering. In the context of $D^{-1}S$, the **Masking Noise** can be interpreted as selectively removing edge

weights or similarity information. The autoencoder is therefore trained to reconstruct missing graph connections from the remaining structure, effectively mapping corrupted node representations $\tilde{x}_i$ to more plausible ones inferred from the global graph topology—namely, the latent community structure. We hypothesize that this denoising process yields node embeddings that are more discriminative for K-MEANS clustering.

3.3 Stacked Architecture Implementation

To create a deep network capable of learning community aware node features, we apply the layer-wise training strategy of stacked denoising autoencoders to the normalized similarity matrix $D^{-1}S$. The procedure is as follows:

1. Train the first layer to denoise the raw input $D^{-1}S$ using the masking noise corruption $q_{\mathcal{D}}$. Note that the corruption of the input data happens repeatedly during training with different corruption patterns (typically at each epoch).
2. Once trained, use the learned encoding function f_θ on *uncorrupted* inputs to generate representations for the next layer.
3. Train the second layer to denoise the representations generated by the first layer.
4. Repeat for T layers and perform K-means on the final output.

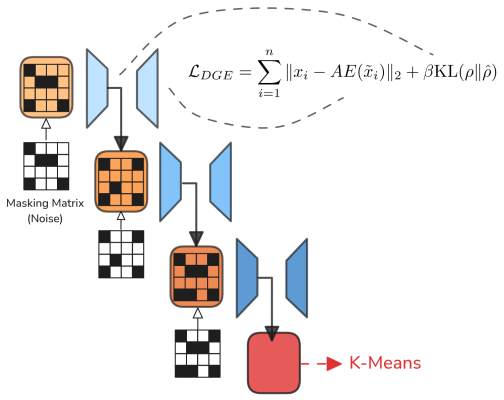

Figure 2: DENOISING GRAPHENCODER

3.4 Benchmarking Denoising GraphEncoder

1. Experimental Setup We focus our evaluation on quantifying the specific contribution of the proposed denoising criterion. To this end, we compare the standard GE directly against our DGE across both the real-world datasets and the LFR synthetic benchmarks used earlier for benchmakring GRAPHENCODER (see subsection 2.4). As previously, we compute both the NCUT and NMI metrics across benchmarks and models. To promote comparison fairness, we train both GE and DGE for a fixed, high number of epochs (determined by analyzing the loss stabilization threshold observed when benchmarking GE, and which was typically below 25000 epochs). We also perform hyper-parameter tuning on DGE using the same experimental setup as for GE (see Appendix B). The reults are presented in Table 2.

Dataset	NMI (Higher is better)		Ncut (Lower is better)	
	Standart GE	DGE	Standard GE	DGE
KARATE	0.837	0.837	0.178	0.178
FOOTBALL	0.924	0.924	3.687	3.687
WINE	0.664	0.664	1.897	1.897
POLBOOKS	0.534	0.586	0.3669	0.316
EMAIL	0.076	0.248	13.946	24.225
LFR ($\mu = 0.01$)	0.974	0.974	0.008	0.008
LFR ($\mu = 0.05$)	0.954	0.954	0.076	0.076
LFR ($\mu = 0.10$)	0.815	0.860	0.343	0.321
LFR ($\mu = 0.20$)	0.130	0.044	0.773	0.755
LFR ($\mu = 0.30$)	0.020	0.036	0.840	0.799
LFR ($\mu = 0.40$)	0.007	0.003	0.812	0.743
LFR ($\mu = 0.50$)	0.004	0.004	0.760	0.745

Table 2: Impact of Denoising: Standard GRAPHENCODER vs. our proposed DENOINSING GRAPHENCODER

2. Analysis of the Results From Table 2, we observe that our DGE almost systematically improved on the NCUT objective (the only exception being EMAIL). This trend that was particularly pronounced across synthetic datasets characterized by low cluster separability—i.e. high μ LFR sets. These findings corroborate our hypothesis that the denoising criterion is well suited for graph clustering, compelling the network to prioritize global community structures over potentially noisy local edge connections.

On the other hand, it is notable that the reduction in NCUT did not strictly correlate with an improvement in the NMI: while we observe improvements on POLBOOKS and LFRs with $\mu = 0.10$ and $\mu = 0.30$, we also note a decrease in NMI for LFR with $\mu = 0.20$ and $\mu = 0.40$. This divergence highlights the distinct natures of these indicators: Ncut is a purely topological objective dependent solely on the adjacency structure of the graph, whereas NMI measures alignment with external semantic ground truth. Consequently, if the graph generation process do not perfectly adhere to the NCut minimization assumption (where edges strictly define class membership), the topological optimum minimized by the network will naturally deviate

from the semantic optimum required by NMI. This is especially visible in the case of the EMAIL benchmark, where DGE had twice as much NCUT as GE, and yet outperformed in NMI by a factor 3.

4 Other Explored Paths

1. Gaussian Mixture Assignment To address the limitations of the standard assignment step, we investigated the use of Gaussian Mixture Models (GMM) as a generalization of the K-means algorithm. Mathematically, the GMM approach is strictly more expressive: while K-means enforces a hard partitioning based on spherical clusters and implicitly assumes a uniform prior on cluster size (equipartition), GMMs operate via probabilistic soft assignment. By optimizing the log-likelihood, GMMs learn full covariance matrices $\mathbf{\Sigma}_k$ and mixing coefficients π_k, allowing them to model clusters with varying ellipsoidal geometries and imbalances in population size. Despite this theoretical advantage, the empirical results (detailed in Annex, Table 4) remained comparable to those obtained via K-means, suggesting that the embedding space learned by the DGE has been regularized into a spherical manifold where the simpler bias of K-means is sufficient. More details can be found in Appendix C.

2. Optimal Transport Regularization To enforce cluster compactness, we tried to introduce an auxiliary Optimal Transport (OT) loss, minimizing the discrete Wasserstein distance between node embeddings Z and learnable centroids C:

$$\mathcal{L}_{OT}(Z, C) = \min_{\Gamma \in \Pi(\mu,\nu)} \sum_{i,j} \Gamma_{ij} \|z_i - c_j\|^2 \tag{6}$$

where Γ is the transport plan. We observed two critical optimization failures that cannot be resolved (see Appendix E for more details):

1. **Non-Stationary Cost Landscape:** Jointly optimizing the encoder parameters θ and centroids C proved intractable. As θ updates, the embedding distribution $\mu(Z_\theta)$ shifts continuously, rendering the cost matrix $M_{ij} = \|z_i - c_j\|^2$ non-stationary. This prevents the centroids from tracking the manifold, leading to oscillation.
2. **Mode Collapse under Relaxation:** To avoid giving a uniform prior of the cluster size, we relaxed the marginal constraint on ν. However, in the absence of repulsive constraints or fixed marginals, the optimization degenerated. The centroids C converged to a single spatial mode (the global data mean) to minimize aggregate transport cost:

$$\lim_{t \to \infty} c_j^{(t)} \approx \mathbb{E}[Z], \quad \forall j \tag{7}$$

3. Vector Quantized-Variational Autoencoders (VQ-VAEs) Finally, we conducted a preliminary investigation into VQ-VAEs (van den Oord et al., 2017). Unlike the current two-stage approach, which sequentially optimizes a continuous embedding followed by discrete k-means clustering, a VQ-VAE framework offers the theoretical advantage of joint optimization. This would allow for the simultaneous learning of the graph embedding and the discrete cluster assignments, potentially resolving the misalignment between the reconstruction objective and the clustering target. Preliminary results can be found in Appendix D.

5 Conclusion

In this work, we reproduced and critically evaluated the GRAPHENCODER method for graph clustering. Our experiments confirm that GRAPHENCODER consistently improves upon direct K-means applied to the normalized similarity matrix. However, in contrast to the original study, we find that GRAPHENCODER does not systematically outperform spectral clustering. Instead, both methods achieve comparable performance across real-world and synthetic benchmarks. We attribute this discrepancy primarily to the sensitivity of GRAPHENCODER to hyperparameter choices and the computational effort required for extensive tuning.

To address these limitations, we introduced the DENOISING GRAPHENCODER, motivated by the manifold hypothesis and designed to generate node embeddings that better reflect global community structure. Empirically, the denoising criterion leads to systematic improvements in the normalized cut objective, particularly on challenging synthetic graphs with weak cluster separability. However, these gains do not always translate into higher normalized mutual information, highlighting the distinction between optimizing purely topological objectives and recovering semantic ground truth.

Overall, our results suggest that autoencoder-based methods constitute a flexible and competitive alternative to spectral clustering, but their practical effectiveness depends strongly on training objectives, evaluation criteria, and computational resources. We believe future work could explore hybrid spectral–deep approaches, cluster-number estimation

Contribution Statement

- **Tristan MARTIN:** Contributed to the architectural refinement and theoretical foundation of the Denoising GraphEncoder (DGE). Provided a rigorous geometric interpretation of the denoising process and established a formal theoretical link between Spectral Clustering and GraphEncoder latent spaces. On the implementation front, enhanced model robustness through the implementation DGE, tested the integration of some Optimal Transport (OT) and integrated GMM clustering for both models and explored VQ-VAE frameworks to improve discrete representation learning.
- **Younes BOUKACEM:** Contributed to the intepretation of the analogy between spectral clustering and autoencoder-based graph clustering, the implementation of the PyTorch replica of GraphEncoder, the implementation of the Optuna hyper-parameter tuning pipeline used for GraphEncoder and Denoising GraphEncoder, the generation of the LFR benchmarks, the scrapping of the real world reduced size benchmarks, and the benchmarking and analysis of GraphEncoder.

References

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 US election. *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*.

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. https://arxiv.org/abs/1907.10902

Chan, P. K., Schlag, M. D. F., & Zien, J. Y. (1993). Spectral k-way ratio-cut partitioning and clustering. *Proceedings of the 30th International Design Automation Conference*, 749–754. https://doi.org/10.1145/157485.165117

Chapelle, O. (2006). Semi-supervised learning.

Dua, D., & Graff, C. (2019). UCI machine learning repository.

Fefferman, C., Mitter, S., & Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, *29*(4), 983–1049. https://doi.org/10.1090/jams/852

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(12), 7821–7826.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, *78*(4). https://doi.org/10.1103/physreve.78.046110

Leskovec, J., Kleinberg, J., & Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 462–471.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 849–856.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. https://arxiv.org/abs/1912.01703

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 833–840.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, *22*(8), 888–905. https://doi.org/10.1109/34.868688

Tian, F., Gao, B., Cui, Q., Chen, E., & Liu, T.-Y. (2014). Learning deep representations for graph clustering. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1293–1299.

van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. *Advances in Neural Information Processing Systems*, *30*, 6306–6315.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, *11*, 3371–3408.

White, S., & Smyth, P. (2005). A spectral clustering approach to finding communities in graphs. *Proceedings of the 2005 SIAM international conference on data mining*, 274–285.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*(4), 452–473.

Appendix

A Derivation of Spectral Clustering and Connection with Autoencoders

A.1 Theoretical Derivation of Spectral Clustering

Spectral clustering provides a principled relaxation of the *Normalized Cut* (NCut) criterion in graph clustering. In this section, we present a rigorous derivation from the combinatorial NCut objective to its continuous spectral optimization form. To this end, let $G = (V, E, W)$ be a weighted undirected graph with $n = |V|$ nodes. Define:

- Adjacency (similarity) matrix $W = [w_{ij}]$.
- Degree matrix $D = \text{diag}(d_1, \ldots, d_n)$, with $d_i = \sum_j w_{ij}$.
- A partition of nodes V into k disjoint clusters $A_1, \ldots, A_k$, such that $\bigcup_i A_i = V$.

using this notation, we can define the *Normalized Cut* objective as:

$$\text{NCut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}, \tag{8}$$

where:

$$\text{cut}(A_i, \bar{A}_i) = \sum_{u \in A_i, v \in \bar{A}_i} w_{uv}, \tag{9}$$

$$\text{vol}(A_i) = \sum_{u \in A_i} d_u. \tag{10}$$

To derive the spectral clustering relaxation, we need to to reformulate the combinatorial optimization problem defined in Equation 8. Let $L = D - W$ be the graph Laplacian. For a cluster A_j, define the *unnormalized indicator vector* $f_j \in \mathbb{R}^n$:

$$(f_j)_i = \begin{cases} 1, & \text{if } v_i \in A_j, \\ 0, & \text{otherwise.} \end{cases}$$

then the Laplacian quadratic form can be expressed as:

$$x^\top L x = \frac{1}{2} \sum_{i,\ell} w_{i\ell}(x_i - x_\ell)^2. \tag{11}$$

Applied to f_j, we obtain:

$$f_j^\top L f_j = \frac{1}{2} \sum_{i,\ell} w_{i\ell}(f_{j,i} - f_{j,\ell})^2. \tag{12}$$

Now, if we consider each pair (i, ℓ):

- If $i, \ell \in A_j$, then $f_{j,i} - f_{j,\ell} = 0$.
- If $i, \ell \notin A_j$, then $f_{j,i} - f_{j,\ell} = 0$.
- If $i \in A_j$ and $\ell \notin A_j$, then $(f_{j,i} - f_{j,\ell})^2 = 1$.

Hence, only edges crossing the cut contribute, giving:

$$f_j^\top L f_j = \sum_{i \in A_j, \ell \in \bar{A}_j} w_{i\ell} = \text{cut}(A_j, \bar{A}_j). \tag{13}$$

For the *Normalized Cut* (NCut), we introduce a volume-normalized indicator:

$$h_j = \frac{f_j}{\sqrt{\text{vol}(A_j)}}, \quad \text{vol}(A_j) = \sum_{i \in A_j} d_i. \tag{14}$$

Then:

$$h_j^\top L h_j = \frac{f_j^\top L f_j}{\text{vol}(A_j)} = \frac{\text{cut}(A_j, \bar{A}_j)}{\text{vol}(A_j)}. \tag{15}$$

Finally, summing over all clusters gives the NCut in matrix form:

$$\mathrm{NCut}(A_1, \ldots, A_k) = \sum_{j=1}^{k} h_j^\top L h_j = \mathrm{Tr}(H^\top L H), \tag{16}$$

where $H = [h_1, \ldots, h_k] \in \mathbb{R}^{n \times k}$ is the matrix of normalized indicators.
When we add the constraints together, we can find the following optimization problem.

$$\begin{aligned} \min_{H \in \mathbb{R}^{n \times k}} \quad & \mathrm{Tr}\big(H^\top L H\big) \\ \text{s.t.} \quad & H^\top D H = I_k, \\ \text{where} \quad & H_{ij} = \begin{cases} \dfrac{1}{\sqrt{\mathrm{vol}(V_j)}}, & \text{if } v_i \in V_j, \\ 0, & \text{otherwise,} \end{cases} \\ & D = \mathrm{diag}(d_1, d_2, \ldots, d_n), \quad L = D - S, \\ & \mathrm{vol}(V_j) = \sum_{v_i \in V_j} d_i. \end{aligned}$$

This problem is **NP-hard**. In order to have a chance to solve that problem, classic spectral method decided to relax the constraint on H in order to be able to use classic solver convex optimization. Specifically, in spectral clustering, H_ij values are allowed to take any real values. Under this relaxation, let $Y = D^{1/2}H$ so that $H = D^{-1/2}Y$. Then:

$$\mathrm{Tr}(H^\top L H) = \mathrm{Tr}(Y^\top D^{-1/2} L D^{-1/2} Y) \tag{17}$$

$$= \mathrm{Tr}(Y^\top L_{\mathrm{sym}} Y), \tag{18}$$

where $L_{\mathrm{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} S D^{-1/2}$. The constraint then becomes $Y^\top Y = I_k$. Hence the relaxed spectral problem is:

$$\min_{Y \in \mathbb{R}^{n \times k}} \mathrm{Tr}(Y^\top L_{\mathrm{sym}} Y) \quad \text{s.t. } Y^\top Y = I_k. \tag{19}$$

By the Ky Fan theorem, the solution is given by the k eigenvectors of L_{sym} corresponding to the *smallest eigenvalues*:

$$Y = [v_1, \ldots, v_k], \quad L_{\mathrm{sym}} v_i = \lambda_i v_i, \quad i = 1, \ldots, k. \tag{20}$$

Finally, the clusters are obtained by running K-Means on the rows of Y. The key steps of this derivation are summarized in Table 3.

Stage	Optimization Problem	Domain	Solution
Discrete NCut	$\min_{H \in \mathcal{I}} \mathrm{Tr}(H^\top L H),\ H^\top D H = I$	Combinatorial	NP-hard
Relaxed Spectral	$\min_{H^\top D H = I} \mathrm{Tr}(H^\top L H)$	Continuous	$H = D^{-1/2} Y$
Normalized Form	$\min_{Y^\top Y = I} \mathrm{Tr}(Y^\top L_{\mathrm{sym}} Y)$	Continuous	$Y =$ eigenvectors of L_{sym}

Table 3: Summary of the key steps in the theoretical derivation of spectral clustering

A.2 Analogy between Autoencoders and Spectral Clustering

A.2.1 Autoencoders and Spectral clustering Equivalence

A linear autoencoder with bottleneck dimension k and reconstruction loss:

$$\mathcal{L}_{recon} = \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2, \tag{21}$$

learns a k-dimensional subspace representation of the input data X. Here, f encodes the input to a low-dimensional latent space and g decodes it back. This is mathematically analogous to spectral clustering: both methods aim to minimize a reconstruction error under a low-rank constraint. To see this, consider the normalized graph Laplacian:

$$Q = D^{-1} L = I - D^{-1} S,$$

The relation between Q and the symmetric normalized Laplacian L_{sym} is the following :

$$L_{\text{sym}} = D^{1/2} Q D^{-1/2}$$

in which case:

$$\text{if } Qu = \lambda u \text{ then } L_{\text{sym}}(D^{1/2}u) = \lambda(D^{1/2}u).$$

meaning that the eigenvalues of both matrices are the same. We will focus now on Q instead of the L_{sym} because the autoencoder uses the $D^{-1}S$ instead of $D^{-1/2}SD^{-1/2}$. We have that the relaxed problem solved by spectral clustering is:

$$\min_{Y \in \mathbb{R}^{n \times k}} \text{Tr}(Y^\top L_{\text{sym}} Y) \quad \text{s.t. } Y^\top Y = I_k. \tag{22}$$

with solution given by the k eigenvectors of L_{sym} corresponding to the k smallest eigenvalues of L_{sym}, which are the same as the eigenvector of Q subject to a matrix multiplication by $D^{1/2}$ associated with the smallest eigenvalue as shown above. Since $Q = I - D^{-1}S$, these correspond to the eigenvector of $D^{-1}S$ corresponding to the largest eigenvalue.

By the Eckart-Young-Mirsky theorem, the best rank-k approximation of a symmetric matrix $D^{-1}S$ under the Frobenius norm is obtained by

$$\min_{\text{rank}(X)=k} \|D^{-1}S - X\|_F = \|D^{-1}S - Y_k \Lambda_k Y_k^T\|_F, \tag{23}$$

where $Y_k \in \mathbb{R}^{n \times k}$ contains the top-k eigenvectors of $D^{-1}S$ and Λ_k their eigenvalues.

Hence, spectral clustering can be interpreted as a linear autoencoder applied to $D^{-1}S$, with Y_k playing the role of the latent embedding, and the reconstruction $Y_k \Lambda_k Y_k^T$ minimizing the matrix reconstruction error. Both methods aim to encode the similarity information in a low-dimensional space that preserves most of the original structure, which justifies the use of autoencoders for graph clustering.

A.2.2 Flexibility of Autoencoders over Spectral Clustering for Graph Clustering

As we have seen, both spectral clustering and autoencoders aim to encode the similarity matrix into a low-dimensional representation that preserves the structure of the graph. However, autoencoders offer significantly greater flexibility in practice, especially for large-scale or noisy graphs.

In spectral clustering, the embeddings are given by the top eigenvectors of the normalized similarity matrix. These eigenvectors are typically dense, which makes it difficult to enforce sparsity—a property often desirable to reduce storage and computational costs and to filter out weak or noisy connections that can harm clustering quality. Adding a sparsity constraint directly to the spectral clustering objective is non-trivial and can compromise the spectral properties that make the method effective.

Autoencoders, on the other hand, allow sparsity to be incorporated naturally. Sparse autoencoders achieve this by augmenting the reconstruction loss with a sparsity regularization term:

$$\mathcal{L}_{SAE} = \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2 + \beta \sum_{j=1}^{d_h} \text{KL}(\rho \| \hat{\rho}_j), \tag{24}$$

where $\hat{\rho}_j$ is the average activation of hidden unit j, ρ is a small target activation, and β controls the strength of the sparsity penalty. This regularization encourages hidden units to activate only selectively, producing embeddings that are both sparse and informative. Importantly, this can be optimized efficiently using standard backpropagation.

Furthermore, stacking multiple layers of sparse autoencoders allows the model to learn hierarchical, non-linear representations of the graph. Each layer progressively refines the embedding, capturing more complex structures and relationships, which in turn improves clustering performance. Also, autoencoders maintain manageable complexity even for large datasets through backpropagation. These advantages make autoencoder-based methods particularly suitable for flexible, scalable, and robust graph clustering.

B Benchmarking Experimental Setup

To evaluate the performance of GraphEncoder on standard graph-clustering benchmarks such as the Karate Club network and LFR-generated graphs, we conduct a systematic hyperparameter optimization using the Optuna framework. The objective of this optimization is to identify an encoder architecture and training configuration that yield low values of the *Normalized Cut* (NCut), which serves as an unsupervised internal clustering criterion. In contrast to metrics such as the Normalized Mutual Information (NMI), NCut does not rely on ground-truth community labels; hence, using it as the optimization objective prevents supervision-induced bias during model selection.

B.1 Search Space

Each Optuna trial corresponds to a full training and evaluation cycle of a candidate GraphEncoder configuration. The following hyperparameters are sampled:

Hidden-layer decay rate. A geometric decay rate $\delta \in [0.6, 0.9]$ controls the successive reduction of hidden dimensions. Starting from the input dimension (equal to the number of nodes), the hidden-layer widths are iteratively computed as

$$d_{k+1} = \lfloor \delta \, d_k \rfloor,$$

until the dimensionality would fall below the number of clusters. This produces a decreasing sequence of layer sizes forming a tapered encoder architecture.

Number of layers. Let $h = [h_1, h_2, \ldots, h_{L_{max}}]$ be the list of consecutive hidden dimensions of the encoder as generated in the previous step. The number of encoder layers is selected by sampling uniformly in $[1..L_{max}]$ a stopping index L in the list, yielding to a set of hidden layers $h' = [h_1, \ldots, h_L]$.

Sparsity parameters. Following the sparsity-inducing formulation of GraphEncoder, the target activation sparsity ρ is sampled from $[10^{-4}, 10^{-1}]$ on a logarithmic scale, and the sparsity penalty weight β from $[10^{-2}, 10^{3}]$ (also log-uniform).

Optimization parameters. The learning rate η of the AdamW optimizer is sampled log-uniformly in $[10^{-3}, 10^{-2}]$.

B.2 Layer-wise Greedy Pretraining

The GraphEncoder is trained using a greedy layer-wise pretraining strategy typical of deep autoencoders. For each layer $\ell \in \{1, \ldots, L\}$:

1. The autoencoder associated with layer ℓ is trained to minimize a loss consisting of (i) the reconstruction error between input and decoder output, and (ii) a Kullback–Leibler divergence term enforcing the sparsity target ρ. In our setup, the training stops once the loss stabilizes in within a 10^{-4} tolerance for 5 consecutive itertions, or the training time reaches a fixed limit (3 minutes).
2. After training layer ℓ, the encoder part of the autoencoder is applied to the training data, producing latent representations that serve as input for layer $\ell + 1$.

Training proceeds for L layers, after which all encoders are composed to obtain the final low-dimensional representation of the graph.

B.3 Evaluation Within Each Trial

After training, the model is evaluated in the following manner:

1. The encoder part of the obtained GraphEncoder is applied to the normalized similarity matrix $D^{-}1S$ of the graph, producing a latent embedding.
2. K-Means is applied to the latent space to otbain the final clustering. To ensure robustness with respect to K-Means initialization, clustering is performed 100 times with different seeds and the best clustering with respect to inertia is returned.
3. We compute both the NCut and the NMI metrics of the obtained clustering.

Optuna uses the NCut as the objective to be minimized for hyper-parameter tuning. In parallel, the study additionally tracks:

- the NMI associated with the trial achieving the lowest NCut. This allows us to report the **ncut-ge** and **nmi-ge** metrics.
- the best NMI observed across all trials, which corresponds to the **best-nmi-ge** metric.

B.4 Optimization Setup

We conduct 20 Optuna trials using a Tree-structured Parzen Estimator (TPE) sampler with a fixed random seed for reproducibility. All experiments use a batch size equal to the number of nodes in the graph.

C Benchmark with Gaussian Mixture Assignments

In the proposed GraphEncoder framework, the final clustering step is traditionally performed using the K-means algorithm. While effective, K-means can be viewed as a restricted limit of a probabilistic model, enforcing strong priors on both the geometry and the cardinality of the clusters.

To generalize the clustering phase and relax these constraints, we explored the use of Gaussian Mixture Models (GMM). Theoretically, GMMs provide a more flexible partitioning of the latent space embedding $H \in \mathbb{R}^{n \times k}$ by modeling the density of the node representations as a weighted sum of K Gaussian component densities:

$$p(\mathbf{h}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{h} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{25}$$

where $\mathbf{h}$ represents a row in the embedding matrix, π_k is the mixing coefficient representing the prior probability of cluster k, and $\mathcal{N}(\mathbf{h} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the multivariate Gaussian density.

Relaxation of the Uniform Prior A critical distinction lies in the treatment of cluster sizes. The standard K-means objective implicitly assumes a uniform prior, where $\pi_1 = \cdots = \pi_K = 1/K$. This imposes a bias toward generating balanced clusters of approximately equal size, often splitting larger natural communities to satisfy this constraint. In contrast, the GMM optimizes the mixing coefficients π_k via the Expectation-Maximization (EM) algorithm. This allows the model to adaptively learn the relative "mass" of each cluster, theoretically enabling the detection of highly imbalanced communities (e.g., a large core cluster coexisting with smaller satellite clusters).

Additionally, GMMs learn full covariance matrices $\boldsymbol{\Sigma}_k$, allowing for ellipsoidal decision boundaries, whereas K-means is restricted to spherical clusters ($\boldsymbol{\Sigma}_k = \mathbf{I}$). Prior to fitting the GMM, we apply L_2 normalization to the embeddings, projecting them onto the unit hypersphere to stabilize EM convergence.

Empirical Observation Despite the theoretical advantages of GMMs in capturing both complex geometries and varying cluster sizes, our experiments yielded Normalized Mutual Information (NMI) scores that were comparable to, though frequently marginally lower than, those obtained via K-means.

This counter-intuitive result suggests a significant property of the GraphEncoder's objective function. The autoencoder, specifically with the imposed sparsity constraints, appears to learn a non-linear embedding where the optimal decision boundaries are inherently spherical and the clusters are naturally balanced. Consequently, the additional degrees of freedom provided by the GMM—specifically the variable covariance and mixing coefficients—may lead to overfitting on this specific manifold, whereas the high bias of K-means acts as a beneficial regularizer. Finally, we must acknowledge that this observation may be an artifact of the specific datasets employed, which may not sufficiently represent the full spectrum of topological diversity and cluster imbalance found in broader classes of complex networks.

Dataset	**NMI (Higher is better)**		**Ncut (Lower is better)**	
	Standard AE	**RobustSAE**	**Standard AE**	**RobustSAE**
Karate	0.837	0.837	0.178	0.178
Football	0.920	**0.923**	3.924	**3.793**
Wine	0.664	0.664	1.897	1.897
Polbooks	**0.559**	0.504	0.431	**0.422**
Email	0.076	**0.248**	**13.946**	24.225
LFR ($\mu = 0.01$)	0.974	0.974	0.008	0.008
LFR ($\mu = 0.05$)	0.954	0.954	0.076	0.076
LFR ($\mu = 0.10$)	**0.694**	0.564	**0.409**	0.658
LFR ($\mu = 0.20$)	0.120	**0.166**	**0.783**	0.807
LFR ($\mu = 0.30$)	0.003	**0.020**	**0.751**	0.839
LFR ($\mu = 0.40$)	**0.007**	0.003	0.812	**0.743**
LFR ($\mu = 0.50$)	0.004	0.004	0.760	**0.745**

Table 4: Impact of Denoising: Standard AE vs. RobustSAE. The cluster assignment is realized with a Gaussian Mixture Model.

D Proposal for Discretized Graph Embedding via VQ-VAE

D.1 Motivation: Bridging the Relaxation Gap

The GraphEncoder method currently operates by learning a non-linear continuous embedding via Stacked Sparse Autoencoders (SAE), followed by k-means clustering (**Tian2014**). While this effectively approximates the normalized graph similarity matrix reconstruction (**Tian2014**), it suffers from a decoupling problem: the embedding is optimized for reconstruction, while the clustering is performed as a separate, subsequent step.

We propose integrating the **Vector Quantized-Variational AutoEncoder (VQ-VAE)** to replace the SAE. Unlike standard VAEs which assume a continuous latent space (typically Gaussian), the VQ-VAE learns a *discrete* latent representation. This architecture is particularly adept for graph clustering as it forces the graph nodes to map directly to a finite set of discrete vectors (codebook), effectively performing clustering and representation learning simultaneously in an end-to-end manner.

D.2 Architecture and Clustering Mechanism

The VQ-VAE consists of three primary components: an encoder, a discrete codebook, and a decoder.

The Encoder and Latent Space: Let $x \in \mathbb{R}^n$ represent a row of the normalized graph similarity matrix $D^{-1}S$ (input data). The encoder network $E(\cdot)$ maps x to a continuous latent vector $z_e(x)$:

$$z_e(x) = E(x; \theta_E) \tag{26}$$

The Codebook (Clustering Centroids): We define a codebook $E = \{e_k\}_{k=1}^K \subset \mathbb{R}^D$, where K is the predefined number of clusters (analogous to k in k-means) and D is the dimensionality of the latent embedding. The discrete latent variable z is calculated by a nearest-neighbor lookup in this codebook. The quantization process $q(\cdot)$ maps $z_e(x)$ to the closest codebook vector e_k:

$$z_q(x) = e_k \quad \text{where} \quad k = \arg\min_j \|z_e(x) - e_j\|_2 \tag{27}$$

In the context of graph clustering, the index k represents the cluster assignment for the node corresponding to input x.

The Decoder: The decoder $G(\cdot)$ reconstructs the input graph signal from the quantized vector $z_q(x)$:

$$\hat{x} = G(z_q(x); \theta_G) \tag{28}$$

D.3 Training and Optimization

The critical innovation of VQ-VAE is handling the non-differentiable $\arg\min$ operation. Training utilizes the *Straight-Through Estimator*, where the gradients from the decoder input $z_q(x)$ are copied directly to the encoder output $z_e(x)$ during backpropagation.

The total objective function combines reconstruction loss (preserving graph structure) with terms to align the codebook and the encoder outputs:

$$\mathcal{L} = \underbrace{\|x - \hat{x}\|_2^2}_{\text{Reconstruction Loss}} + \underbrace{\|sg[z_e(x)] - e\|_2^2}_{\text{Codebook Loss}} + \underbrace{\beta \|z_e(x) - sg[e]\|_2^2}_{\text{Commitment Loss}} \tag{29}$$

where $sg[\cdot]$ denotes the stop-gradient operator and β is a hyperparameter.

- **Reconstruction Loss:** Ensures the discrete embedding captures the essential topology of the graph (similar to Eq. 5 in GraphEncoder **Tian2014**).
- **Codebook Loss:** Moves the embedding vectors e_k (centroids) towards the encoder outputs $z_e(x)$.
- **Commitment Loss:** Prevents the encoder outputs $z_e(x)$ from fluctuating too wildly between codebook vectors, ensuring stable clustering.

D.4 Expected Improvements over GraphEncoder

By replacing SAE with VQ-VAE, we transition from soft, continuous constraints (sparsity penalties) to hard, discrete constraints (vector quantization). This implies: 1. **End-to-End Clustering:** The cluster assignment is learned jointly with the reconstruction, trying to ensure the embedding is optimal for partitioning. 2. **Discrete Bottleneck:** Limits the capacity of the model to only store categorical information, naturally filtering noise in the graph adjacency matrix.

D.5 Preliminary Empirical Evaluation

To validate the feasibility of the proposed VQ-VAE architecture, we conducted a preliminary experiment on two graph datasets. The model was trained end-to-end using the discrete bottleneck objective described in Section A.3.

Table 5 summarizes the clustering performance in terms of Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Normalized Cut (Ncut).

Table 5: Preliminary Clustering Results with Graph VQ-VAE

Dataset	NMI	ARI	Ncut
Email	0.5358	0.1862	20.8750
Wine	**0.7969**	**0.8149**	**1.8985**

Analysis of Wine Dataset: On the *Wine* dataset, the VQ-VAE achieves an NMI of **0.7969**. This result is highly competitive with the original GraphEncoder method, which reported an NMI of 0.63 using Stacked Sparse Autoencoders (SAE), and significantly outperforms the spectral clustering baseline (NMI 0.72).

Notably, the VQ-VAE achieves this without a separate k-means step; the discrete codebook naturally partitions the data. The low Ncut value (1.898) confirms that the learned discrete representations effectively capture the graph's community structure, minimizing the cut cost end-to-end. For the Email dataset, the NMI is still slightly higher, however the NCut is also higher, showing the inconsistency between the two indicators.

E Theoretical Analysis of Optimization Instabilities in Joint OT Clustering

In Section 4, we reported that a naive application of Optimal Transport (OT) regularization—specifically, the joint optimization of encoder parameters Θ and unconstrained centroids M—resulted in optimization divergence and mode collapse. Here, we provide a rigorous mathematical derivation of these pathologies, demonstrating that they are fundamental consequences of the optimization landscape in coupled representation learning. So, they could have been partially avoided by a more rigourous technical analysis;

E.1 Pathology I: Gradient Coupling and Non-Stationary Costs

The fundamental instability observed during the simultaneous optimization of the encoder f_Θ and the transport plan stems from the dependency of the ground cost matrix on the learnable parameters Θ.

Let $C(\Theta) \in \mathbb{R}^{n \times k}$ be the cost matrix where $C_{ij}(\Theta) = \|f_\Theta(x_i) - \mu_j\|^2$. In a naive joint optimization framework, we attempt to minimize:

$$\min_{\Theta, M} \mathcal{L}_{total} = \mathcal{L}_{recon}(\Theta) + \gamma \mathcal{W}_2^2(\alpha_\Theta, \beta_M) \tag{30}$$

where the Wasserstein distance $\mathcal{W}_2^2$ involves solving the dual potential maximization or primal transport minimization. Crucially, the gradient of the OT loss with respect to Θ involves the optimal transport plan $\pi^\star$:

$$\nabla_\Theta \mathcal{L}_{OT} = \sum_{i,j} \pi^\star_{ij}(\Theta, M) \cdot \nabla_\Theta \|z_i(\Theta) - \mu_j\|^2 \tag{31}$$

The intractability arises because $\pi^\star$ is the solution to a linear program (or a Sinkhorn projection) parameterized by $C(\Theta)$. As Θ updates, the manifold of embeddings Z warps continuously. However, the optimal transport vertex $\pi^\star$ often exhibits discontinuous jumps when the cost structure changes (the "bang-bang" property of linear programming solutions).

Consequently, the centroids M chase a moving target distribution α_Θ whose geometry shifts faster than the centroids can converge. This results in the oscillatory behavior observed.

E.2 Pathology II: Mode Collapse in Unbalanced Transport Do not know if it is useful

We observed that relaxing the marginal constraints on the centroids (to avoid enforcing a uniform prior on cluster sizes) led to degenerate solutions where all centroids converged to the global data mean $\mathbb{E}[Z]$.

Mathematically, relaxing the target marginal constraint $\sum_i \pi_{ij} = w_j$ transforms the problem from Balanced Optimal Transport into a variant of *Unbalanced Optimal Transport* or simple soft-clustering. If the weights w are learnable and unconstrained, the objective simplifies to:

$$\min_{M, \pi \geq 0} \sum_{i,j} \pi_{ij} \|z_i - \mu_j\|^2 \quad \text{s.t.} \quad \sum_j \pi_{ij} = \frac{1}{n} \tag{32}$$

Without the repulsive force provided by the constraint that each centroid must capture a fixed mass w_j, the optimization seeks to minimize the aggregate transport cost globally. By Jensen's inequality, the transport cost is strictly minimized when all centroids collapse to the single spatial mode that minimizes the variance of the dataset:

$$\lim_{t \to \infty} \mu_j^{(t)} = \frac{1}{n} \sum_{i=1}^{n} z_i, \quad \forall j \in \{1, \ldots, k\} \tag{33}$$

This degeneracy renders the clustering partition vacuous, as $P(c_j|z_i)$ becomes uniform across j.