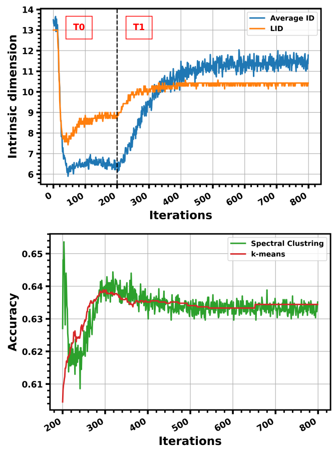
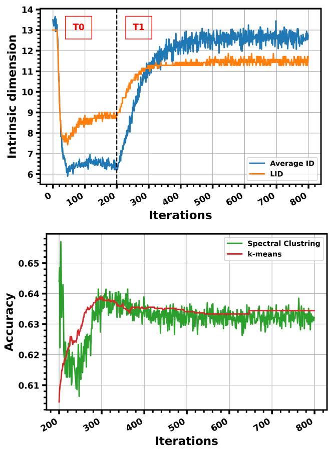
Supplementary Material:
"Escaping Feature Twist: A Variational Graph Auto-Encoder for Node Clustering"

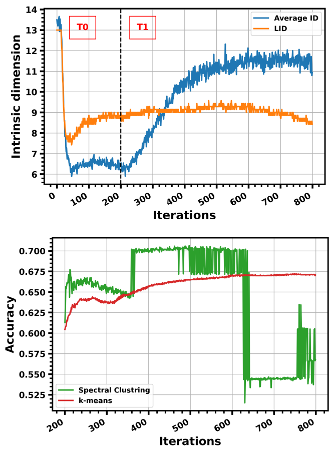
Appendix A: Establishing the existence of Feature Twist



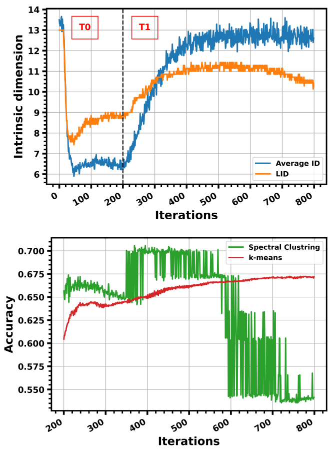
(a) L_c



(b) $L_c + L_r$

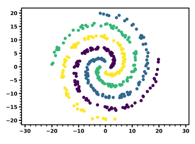


(c) $\mathcal{A}(L_c, L_o)$

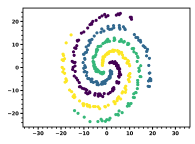


(d) $\mathcal{A}(L_c + L_r, L_o + L_r)$

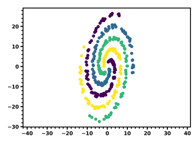
Figure 1: **First evidence of Feature Twist**. Four GAE models sharing the same architecture (2 GCN layers of dimensions $32 - 16$ and activated by ReLU), the same pretraining phase (adjacency reconstruction for 200 epochs), and the same optimizer (Adam with a learning rate equal to 10^{-3}). Each column represents a model. The difference between the four models is limited to the objective function of the clustering phase: (a), (b), (c), and (d). First row: evaluating two geometric properties of the latent manifolds. Second row: evaluating accuracy by applying k-means and spectral clustering on the latent representations. Average ID: average ID of the clustering manifolds. LID: number of dimensions that can capture 90% of the covariance matrix (linear correlations) estimated based on PCA (Principal Component Analysis). T_0: pretraining phase. T_1: clustering phase. L_c: embedded clustering objective (k-means with 7 centers) [Xie *et al.*, 2016]. L_r: reconstruction objective. L_o: embedded over-clustering objective (k-means with 300 centers). $\mathcal{A}(L_1, L_2)$: alternating between L_1 and L_2.



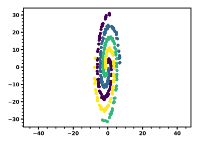
(a) Input data



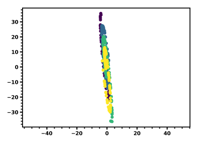
(b) Epoch 200



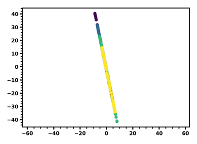
(c) Epoch 225



(d) Epoch 250



(e) Epoch 275



(f) Epoch 300

Figure 2: **Second evidence of Feature Twist**. Collapse of the latent structures in the clustering phase.

In this appendix, we establish the existence of the Feature Twist problem. To this end, we conduct the first set of experiments on Cora, Citeseer, and Pubmed [Sen *et al.*, 2008]. Results on Cora are illustrated in Figure 1. Results on Citeseer and Pubmed exhibit the same patterns for their Intrinsic Dimension (ID) and Linear Intrinsic Dimension (LID) as obtained for Cora, which makes the following analysis valid for the three considered datasets. As we can see, the average ID and LID evolve almost identically for the first few iterations of the pretraining phase. After that, a clear gap between both curves gradually takes place. This result indicates that the pretraining strategy starts by learning linear correlations and then transforms the initial flat manifolds into *curved* ones. Thus, we expect that the Euclidean geometry is inappropriate to assess the latent similarities at the end of the pretraining phase. Without prior knowledge, it is not possible to systematically identify a non-euclidean metric that can capture the latent similarities for any dataset. Furthermore, the discrepancy between the latent space dimension (i.e., 16) and the average ID at the end of the pretraining phase confirms that the latent manifolds are *low-dimensional* (much lower than the dimension of the latent space 16).

In the clustering phase, we observe that the average ID increases considerably and reaches higher values than LID for the four cases studied in Figure 1. This result implies that the embedded manifolds undergo substantial transformation: from curved low-dimensional to flattened higher-dimensional manifolds. This transformation can bring geometric deterioration caused by twisting the curved structures while flattening the latent manifolds as we will show on a 2D synthetic dataset. From the first experiment (Figure 1.a), we can see that spectral clustering yields better results than k-means at the beginning of the clustering phase. After that, we observe a major collapse in the spectral clustering results. At the end, both algorithms (k-means and spectral clustering) perform similarly. This collapse is caused by the inappropriate flattening of the latent manifolds (ID increases to reach LID). From Figure 1.b, we can see that combining clustering and reconstruction can not alleviate the twisting effect (there is still a major collapse in the spectral clustering results when ID increases to reach LID). From Figure 1.c, we observe a slower increase in ID compared with the first experiment (transition less coarse). Additionally, the spectral clustering accuracy reaches higher values. Hence, we can conclude that alternating between clustering and over-clustering flattens the latent manifolds more appropriately (less twisting effect) than the first and second cases (Figure 1.a and Figure 1.b). From Figure 1.d, we observe that adding the reconstruction function does not help in alleviating the twisting effect even when we alternate between clustering and over-clustering (similar spectral clustering results with Figure 1.c).

Since it is not possible to visualise the curved latent structures for the first set of experiments due to the space dimensionality, we conduct a second experiment on a 2D synthetic dataset with four curved clusters. For this experiment, we leverage a linear two-layer auto-encoder that projects the data in a 2D latent space. We pretrain using reconstruction for 200 iterations and fine-tune using a clustering objective [Xie *et al.*, 2016]. The embedded space visualisations are illustrated in Figure 2. As we can see, the clustering phase flattens the curved manifolds. We observe a geometric deterioration caused by twisting the curved structures while flattening the latent manifolds (what we call Feature Twist). This coarse flattening enforces samples from initially separate clusters to form inappropriate flat manifolds and hence degenerate the clustering structures.

Appendix B: Derivation of $\mathcal{L}_1$ in Eq. (8)

We prove that $\mathcal{L}_1$ (as stated below) is a lower bound of the input graph log-likelihood:

$$\mathcal{L}_1 = \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j \sim q(.|X,A)} \Big[\log\big(p(a_{ij}|z_i,z_j)\big) \Big] - 2\,N \sum_{i=1}^{N} KL\big(q(z_i|X,A)||p(z_i)\big).$$

$$\begin{aligned}
\log\big(p(A)\big) &= \log\big(\prod_{i,j=1}^{N} p(a_{ij})\big),\\
&= \sum_{i,j=1}^{N} \log\big(p(a_{ij})\big),\\
&= \sum_{i,j=1}^{N} \log\big(\int_{z_i}\int_{z_j} p(a_{ij},z_i,z_j)\,dz_i\,dz_j\big),\\
&= \sum_{i,j=1}^{N} \log\Big(\int_{z_i}\int_{z_j} \frac{p(a_{ij}|z_i,z_j)\,p(z_i,z_j)}{q(z_i,z_j|X,A)}\,q(z_i,z_j|X,A)\,dz_i\,dz_j\Big),\\
&= \sum_{i,j=1}^{N} \log\Big(\mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\frac{p(a_{ij}|z_i,z_j)\,p(z_i)\,p(z_j)}{q(z_i|X,A)\,q(z_j|X,A)}\Big]\Big),\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(\frac{p(a_{ij}|z_i,z_j)\,p(z_i)\,p(z_j)}{q(z_i|X,A)\,q(z_j|X,A)}\Big)\Big], \quad \text{(Jensen's inequality)}\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big) + \log\Big(\frac{p(z_i)}{q(z_i|X,A)}\Big) + \log\Big(\frac{p(z_j)}{q(z_j|X,A)}\Big)\Big],
\end{aligned}$$

$$\begin{aligned}
\log\big(p(A)\big) &\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] + 2\,N\,\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(.|X,A)}\Big[\log\Big(\frac{p(z_i)}{q(z_i|X,A)}\Big)\Big],\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] - 2\,N\,\sum_{i=1}^{N} KL\Big(q(z_i|X,A)||p(z_i)\Big),\\
&\geqslant \mathcal{L}_1.
\end{aligned}$$

The first term of $\mathcal{L}_1$ can be estimated based on Monte Carlo sampling (L^2 samples) and the reparameterization trick with a complexity $\mathcal{O}(dL^2N^2)$ as follows:

$$\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] \simeq \frac{1}{L^2}\sum_{l_1,l_2=1}^{L}\sum_{i,j=1}^{N}\log\Big(p(a_{ij}|z_i^{(l_1)},z_j^{(l_2)})\Big),$$

$$\begin{aligned}
\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] \simeq\ & \frac{1}{L^2}\sum_{l_1,l_2=1}^{L}\sum_{i,j=1}^{N} a_{ij}\,\log\Big(\text{Sigmoid}\Big((z_i^{(l_1)})^T z_j^{(l_2)}\Big)\Big)\\
&+\frac{1}{L^2}\sum_{l_1,l_2=1}^{L}\sum_{i,j=1}^{N}(1-a_{ij})\,\log\Big(1-\text{Sigmoid}\Big((z_i^{(l_1)})^T z_j^{(l_2)}\Big)\Big).
\end{aligned}$$

The second term of $\mathcal{L}_1$ can be computed analytically with a complexity $\mathcal{O}(dN)$ as follows:

$$2\,N\,\sum_{i=1}^{N} KL\Big(q(z_i|X,A)||p(z_i)\Big) = N\,\sum_{i=1}^{N}\sum_{j=1}^{d}\Big(\mu_{z_i}^2[j]+\sigma_{z_i}^2[j]-\log(\sigma_{z_i}^2[j])-1\Big).$$

Appendix C: Derivation of $\mathcal{L}_2$ in Eq. (10)

We prove that $\mathcal{L}_2$ (as stated below) is a lower bound of the input graph log-likelihood:

$$\mathcal{L}_2 = \frac{1}{N_m}\sum_{i,j=1}^{N}\sum_{l=1}^{N_m}\mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] - 2\,\frac{N}{N_m}\sum_{i=1}^{N}\sum_{l=1}^{N_m} KL\Big(q(\lambda_l(z_i)|X,A)||p(z_i)\Big).$$

$$\begin{aligned}
\log\big(p(A)\big) &= \log\big(\prod_{i,j=1}^{N} p(a_{ij})\big),\\
&= \sum_{i,j=1}^{N}\log\big(p(a_{ij})\big),\\
&= \sum_{i,j=1}^{N}\log\big(\int_{z_i}\int_{z_j} p(a_{ij},z_i,z_j)\,dz_i\,dz_j\big),\\
&= \sum_{i,j=1}^{N}\log\big(\int_{z_i}\int_{z_j} p(a_{ij}|z_i,z_j)\,p(z_i,z_j)\,dz_i\,dz_j\big),\\
&= \sum_{i,j=1}^{N}\log\Big(\int_{z_i}\int_{z_j}\frac{1}{N_m}\sum_{l=1}^{N_m}\frac{p(a_{ij}|z_i,z_j)\,p(z_i)\,p(z_j)}{q(\lambda_l(z_i)|X,A)\,q(\lambda_l(z_j)|X,A)}\,q(\lambda_l(z_i)|X,A)\,q(\lambda_l(z_j)|X,A)\,dz_i\,dz_j\Big),\\
&= \sum_{i,j=1}^{N}\log\Big(\frac{1}{N_m}\sum_{l=1}^{N_m}\mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\frac{p(a_{ij}|z_i,z_j)\,p(z_i)\,p(z_j)}{q(\lambda_l(z_i)|X,A)\,q(\lambda_l(z_j)|X,A)}\Big]\Big),
\end{aligned}$$

$$\begin{aligned}
\log\big(p(A)\big) &= \sum_{i,j=1}^{N} \log\Big(\frac{1}{N_m}\sum_{l=1}^{N_m} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\frac{p(a_{ij}|z_i,z_j)\,p(z_i)\,p(z_j)}{q(\lambda_l(z_i)|X,A)\,q(\lambda_l(z_j)|X,A)}\Big]\Big),\\
&\geqslant \frac{1}{N_m}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(\frac{p(a_{ij}|z_i,z_j)\,p(z_i)\,p(z_j)}{q(\lambda_l(z_i)|X,A)\,q(\lambda_l(z_j)|X,A)}\Big)\Big], \quad \text{(Jensen's inequality)}\\
&\geqslant \frac{1}{N_m}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)+\log\Big(\frac{p(z_i)}{q(\lambda_l(z_i)|X,A)}\Big)+\log\Big(\frac{p(z_j)}{q(\lambda_l(z_j)|X,A)}\Big)\Big],\\
&\geqslant \frac{1}{N_m}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big]+\frac{N}{N_m}\sum_{l=1}^{N_m}\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(\frac{p(z_i)}{q(\lambda_l(z_i)|X,A)}\Big)\Big]\\
&\frac{N}{N_m}\sum_{l=1}^{N_m}\sum_{j=1}^{N}\mathbb{E}_{z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(\frac{p(z_j)}{q(\lambda_l(z_j)|X,A)}\Big)\Big],\\
&\geqslant \frac{1}{N_m}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big]+2\frac{N}{N_m}\sum_{l=1}^{N_m}\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(\frac{p(z_i)}{q(\lambda_l(z_i)|X,A)}\Big)\Big],\\
&\geqslant \frac{1}{N_m}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big]-2\,\frac{N}{N_m}\sum_{l=1}^{N_m}\sum_{i=1}^{N}KL\Big(q(\lambda_l(z_i)|X,A)||p(z_i)\Big),\\
&\geqslant \mathcal{L}_2.
\end{aligned}$$

The first term of $\mathcal{L}_2$ can be estimated based on Monte Carlo sampling (L^2 samples) and the reparameterization trick with a complexity $\mathcal{O}(dN_mL^2N^2)$ as follows:

$$\begin{aligned}
\frac{1}{N_m}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(\lambda_l(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] &\simeq \frac{1}{N_m\,L^2}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N}\sum_{l_1,l_2=1}^{L}\log\Big(p(a_{ij}|z_i^{(l_1)},z_j^{(l_2)})\Big),\\
&\simeq \frac{1}{N_m\,L^2}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N}\sum_{l_1,l_2=1}^{L} a_{ij}\log\Big(\text{Sigmoid}\Big((\lambda_l(z_i)^{(l_1)})^T\lambda_l(z_j)^{(l_2)}\Big)\Big)\\
&+\frac{1}{N_m\,L^2}\sum_{l=1}^{N_m}\sum_{i,j=1}^{N}\sum_{l_1,l_2=1}^{L}(1-a_{ij})\log\Big(1-\text{Sigmoid}\Big((\lambda_l(z_i)^{(l_1)})^T\lambda_l(z_j)^{(l_2)}\Big)\Big).
\end{aligned}$$

The second term of $\mathcal{L}_2$ can be computed analytically with a complexity $\mathcal{O}(dN_mN)$ as follows:

$$2\,\frac{N}{N_m}\sum_{l=1}^{N_m}\sum_{i=1}^{N}KL\Big(q(\lambda_l(z_i)|X,A)||p(z_i)\Big)=\frac{N}{N_m}\sum_{l=1}^{N_m}\sum_{i=1}^{N}\sum_{j=1}^{d}\Big(\mu^2_{\lambda_l(z_i)}[j]+\sigma^2_{\lambda_l(z_i)}[j]-\log(\sigma^2_{\lambda_l(z_i)}[j])-1\Big).$$

Appendix D: Derivation of $\mathcal{L}_3$ in Eq. (16)

We prove that $\mathcal{L}_3$ (as stated below) is a lower bound of the input graph log-likelihood:

$$\mathcal{L}_3=\sum_{i,j=1}^{N}\mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\big(p(a_{ij}|z_i,z_j)\big)\Big]-2N\sum_{i=1}^{N}KL\big(q(z_i|X,A)||p(z_i)\big)-2N\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(.|X,A)}\Big[KL\Big(q(o_i|z_i)||p(o_i|z_i)\Big)\Big].$$

$$
\begin{aligned}
\log\big(p(A)\big) &= \log\big(\prod_{i,j=1}^{N} p(a_{ij})\big),\\
&= \sum_{i,j=1}^{N} \log\big(p(a_{ij})\big),\\
&= \sum_{i,j=1}^{N} \log\big(\sum_{o_i}\sum_{o_j}\int_{z_i}\int_{z_j} p(a_{ij}, z_i, o_i, z_j, o_j)\, dz_i\, dz_j\big),\\
&= \sum_{i,j=1}^{N} \log\big(\sum_{o_i}\sum_{o_j}\int_{z_i}\int_{z_j} p(a_{ij}|z_i, z_j)\, p(o_i|z_i)\, p(o_j|z_j)\, p(z_i)\, p(z_j)\, dz_i\, dz_j\big),\\
&= \sum_{i,j=1}^{N} \log\Big(\sum_{o_i}\sum_{o_j}\int_{z_i}\int_{z_j} \frac{p(a_{ij}|z_i, z_j)\, p(o_i|z_i)\, p(o_j|z_j)\, p(z_i)\, p(z_j)}{q(z_i, o_i, z_j, o_j|X, A)}\, q(z_i, o_i, z_j, o_j|X, A)\, dz_i\, dz_j\Big),\\
&= \sum_{i,j=1}^{N} \log\Big(\sum_{o_i}\sum_{o_j}\int_{z_i}\int_{z_j} \frac{p(a_{ij}|z_i, z_j)\, p(o_i|z_i)\, p(o_j|z_j)\, p(z_i)\, p(z_j)}{q(z_i|X, A)\, q(o_i|z_i)\, q(z_j|X, A)\, q(o_j|z_j)}\, q(z_i|X, A)\, q(o_i|z_i)\, q(z_j|X, A)\, q(o_j|z_j)\, dz_i\, dz_j\Big),\\
&= \sum_{i,j=1}^{N} \log\Big(\mathbb{E}_{\substack{o_i\sim q(.|z_i),\\ o_j\sim q(.|z_j),\\ z_i, z_j\sim q(.|X,A)}} \Big[\frac{p(a_{ij}|z_i, z_j)\, p(o_i|z_i)\, p(o_j|z_j)\, p(z_i)\, p(z_j)}{q(z_i|X, A)\, q(o_i|z_i)\, q(z_j|X, A)\, q(o_j|z_j)}\Big]\Big),\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{\substack{o_i\sim q(.|z_i),\\ o_j\sim q(.|z_j),\\ z_i, z_j\sim q(.|X,A)}} \Big[\log\Big(\frac{p(a_{ij}|z_i, z_j)\, p(o_i|z_i)\, p(z_i)\, p(o_j|z_j)\, p(z_j)}{q(o_i|z_i)\, q(z_i|X, A)\, q(o_j|z_j)\, q(z_j|X, A)}\Big)\Big], \quad \text{(Jensen's inequality)}\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{\substack{o_i\sim q(.|z_i),\\ o_j\sim q(.|z_j),\\ z_i, z_j\sim q(.|X,A)}} \Big[\log\Big(p(a_{ij}|z_i, z_j)\Big) + \log\Big(\frac{p(z_i)}{q(z_i|X, A)}\Big) + \log\Big(\frac{p(z_j)}{q(z_j|X, A)}\Big) + \log\Big(\frac{p(o_i|z_i)}{q(o_i|z_i)}\Big) + \log\Big(\frac{p(o_j|z_j)}{q(o_j|z_j)}\Big)\Big],\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i, z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i, z_j)\Big)\Big] + N\sum_{i=1}^{N} \mathbb{E}_{z_i\sim q(.|X,A)}\Big[\log\Big(\frac{p(z_i)}{q(z_i|X, A)}\Big)\Big]\\
&+ N\sum_{j=1}^{N} \mathbb{E}_{z_j\sim q(.|X,A)}\Big[\log\Big(\frac{p(z_j)}{q(z_j|X, A)}\Big)\Big] + N\sum_{i=1}^{N} \mathbb{E}_{\substack{o_i\sim q(.|z_i),\\ z_i\sim q(.|X,A)}}\Big[\log\Big(\frac{p(o_i|z_i)}{q(o_i|z_i)}\Big)\Big] + N\sum_{i=1}^{N} \mathbb{E}_{\substack{o_j\sim q(.|z_j),\\ z_j\sim q(.|X,A)}}\Big[\log\Big(\frac{p(o_j|z_j)}{q(o_j|z_j)}\Big)\Big],\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i, z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i, z_j)\Big)\Big] + 2N\sum_{i=1}^{N} \mathbb{E}_{z_i\sim q(.|X,A)}\Big[\log\Big(\frac{p(z_i)}{q(z_i|X, A)}\Big)\Big]\\
&+ 2N\sum_{i=1}^{N} \mathbb{E}_{\substack{o_i\sim q(.|z_i),\\ z_i\sim q(.|X,A)}}\Big[\log\Big(\frac{p(o_i|z_i)}{q(o_i|z_i)}\Big)\Big],\\
&\geqslant \sum_{i,j=1}^{N} \mathbb{E}_{z_i, z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i, z_j)\Big)\Big] - 2N\sum_{i=1}^{N} KL\Big(q(z_i|X, A)||p(z_i)\Big)\\
&- 2N\sum_{i=1}^{N} \mathbb{E}_{z_i\sim q(.|X,A)}\Big[KL\Big(q(o_i|z_i)||p(o_i|z_i)\Big)\Big],\\
&\geqslant \mathcal{L}_3.
\end{aligned}
$$

The first term of $\mathcal{L}_3$ can be estimated based on Monte Carlo sampling (L^2 samples) and the reparameterization trick with a complexity $\mathcal{O}(dL^2N^2)$ as follows:

$$\begin{aligned}\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j\sim q(.|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] &\simeq \frac{1}{L^2}\sum_{l_1,l_2=1}^{L}\sum_{i,j=1}^{N}\log\Big(p(a_{ij}|z_i^{(l_1)},z_j^{(l_2)})\Big),\\ &\simeq \frac{1}{L^2}\sum_{l_1,l_2=1}^{L}\sum_{i,j=1}^{N} a_{ij}\log\Big(\text{Sigmoid}\Big((z_i^{(l_1)})^T z_j^{(l_2)}\Big)\Big)\\ &+ \frac{1}{L^2}\sum_{l_1,l_2=1}^{L}\sum_{i,j=1}^{N}(1-a_{ij})\log\Big(1-\text{Sigmoid}\Big((z_i^{(l_1)})^T z_j^{(l_2)}\Big)\Big).\end{aligned}$$

The second term of $\mathcal{L}_3$ can be computed analytically with a complexity $\mathcal{O}(dN)$ as follows:

$$2\,N\sum_{i=1}^{N} KL\Big(q(z_i|X,A)||p(z_i)\Big) = N\sum_{i=1}^{N}\sum_{j=1}^{d}\Big(\mu_{z_i}^2[j]+\sigma_{z_i}^2[j]-\log(\sigma_{z_i}^2[j])-1\Big).$$

The third term of $\mathcal{L}_3$ can be estimated based on Monte Carlo sampling (L samples) and the reparameterization trick with a complexity $\mathcal{O}(LN_oN)$ as follows:

$$\begin{aligned}2\,N\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(.|X,A)}\Big[KL\Big(q(o_i|z_i)||p(o_i|z_i)\Big)\Big] &= 2\,N\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(.|X,A)}\Big[\sum_{o_i} q(o_i|z_i)\log\Big(\frac{q(o_i|z_i)}{p(o_i|z_i)}\Big)\Big],\\ &\simeq 2\,\frac{N}{L}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{o_i} q(o_i|z_i^{(l)})\log\Big(\frac{q(o_i|z_i^{(l)})}{p(o_i|z_i^{(l)})}\Big).\end{aligned}$$

Appendix E: Derivation of $\mathcal{L}_4$ in Eq. (20)

We prove that $\mathcal{L}_4$ (as stated below) is a lower bound of the input graph log-likelihood:

$$\mathcal{L}_4 = \sum_{i,j=1}^{N}\mathbb{E}_{z_i,z_j\sim q(\gamma(.)|X,A)}\Big[\log\Big(p(a_{ij}|z_i,z_j)\Big)\Big] - 2N\sum_{i=1}^{N} KL\Big(q(\gamma(z_i)|X,A)||p(z_i)\Big) - 2N\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(.|X,A)}\Big[KL\Big(q(c_i|z_i)||p(c_i|z_i)\Big)\Big].$$

$$\begin{aligned}\log\big(p(A)\big) &= \log\big(\prod_{i,j=1}^{N} p(a_{ij})\big),\\ &= \sum_{i,j=1}^{N}\log\big(p(a_{ij})\big),\\ &= \sum_{i,j=1}^{N}\log\big(\sum_{c_i}\sum_{c_j}\int_{z_i}\int_{z_j} p(a_{ij},z_i,c_i,z_j,c_j)\,dz_i\,dz_j\big),\\ &= \sum_{i,j=1}^{N}\log\big(\sum_{c_i}\sum_{c_j}\int_{z_i}\int_{z_j} p(a_{ij}|z_i,z_j)\,p(c_i|z_i)\,p(c_j|z_j)\,p(z_i)\,p(z_j)\,dz_i\,dz_j\big),\\ &= \sum_{i,j=1}^{N}\log\Big(\sum_{c_i}\sum_{c_j}\int_{z_i}\int_{z_j}\frac{p(a_{ij}|z_i,z_j)p(c_i|z_i)p(c_j|z_j)p(z_i)p(z_j)}{q(\gamma(z_i)|X,A)q(c_i|z_i)q(\gamma(z_j)|X,A)q(c_j|z_j)}q(\gamma(z_i)|X,A)q(c_i|z_i)q(\gamma(z_j)|X,A)q(c_j|z_j)dz_idz_j\Big),\\ &= \sum_{i,j=1}^{N}\log\Big(\mathbb{E}_{\substack{c_i\sim q(.|z_i),\\ c_j\sim q(.|z_j),\\ z_i,z_j\sim q(\gamma(.)|X,A)}}\Big[\frac{p(a_{ij}|z_i,z_j)\,p(c_i|z_i)\,p(c_j|z_j)\,p(z_i)\,p(z_j)}{q(\gamma(z_i)|X,A)\,q(c_i|z_i)\,q(\gamma(z_j)|X,A)\,q(c_j|z_j)}\Big]\Big),\\ &\geqslant \sum_{i,j=1}^{N}\mathbb{E}_{\substack{c_i\sim q(.|z_i),\\ c_j\sim q(.|z_j),\\ z_i,z_j\sim q(\gamma(.)|X,A)}}\Big[\log\Big(\frac{p(a_{ij}|z_i,z_j)\,p(c_i|z_i)\,p(z_i)\,p(c_j|z_j)\,p(z_j)}{q(c_i|z_i)\,q(\gamma(z_i)|X,A)\,q(c_j|z_j)\,q(\gamma(z_j)|X,A)}\Big)\Big],\quad \text{(Jensen's inequality)}\end{aligned}$$

$$
\begin{aligned}
\log\Big(p(A)\Big) \geqslant & \sum_{i,j=1}^{N} \mathbb{E}_{\substack{c_i \sim q(.|z_i),\\ c_j \sim q(.|z_j),\\ z_i,z_j \sim q(\gamma(.)|X,A)}} \left[\log\Big(\frac{p(a_{ij}|z_i,z_j)\, p(c_i|z_i)\, p(z_i)\, p(c_j|z_j)\, p(z_j)}{q(c_i|z_i)\, q(\gamma(z_i)|X,A)\, q(c_j|z_j)\, q(\gamma(z_j)|X,A)}\Big)\right],\\
\geqslant & \sum_{i,j=1}^{N} \mathbb{E}_{\substack{c_i \sim q(.|z_i),\\ c_j \sim q(.|z_j),\\ z_i,z_j \sim q(\gamma(.)|X,A)}} \left[\log\Big(p(a_{ij}|z_i,z_j)\Big) + \log\Big(\frac{p(z_i)}{q(\gamma(z_i)|X,A)}\Big) + \log\Big(\frac{p(z_j)}{q(\gamma(z_j)|X,A)}\Big) + \log\Big(\frac{p(c_i|z_i)}{q(c_i|z_i)}\Big)\right.\\
& \left. + \log\Big(\frac{p(c_j|z_j)}{q(c_j|z_j)}\Big)\right],\\
\geqslant & \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j \sim q(\gamma(.)|X,A)} \left[\log\Big(p(a_{ij}|z_i,z_j)\Big)\right] + N\sum_{i=1}^{N} \mathbb{E}_{z_i \sim q(\gamma(.)|X,A)} \left[\log\Big(\frac{p(z_i)}{q(\gamma(z_i)|X,A)}\Big)\right]\\
& + N\sum_{j=1}^{N} \mathbb{E}_{z_j \sim q(\gamma(.)|X,A)} \left[\log\Big(\frac{p(z_j)}{q(\gamma(z_j)|X,A)}\Big)\right] + N\sum_{i=1}^{N} \mathbb{E}_{\substack{c_i \sim q(.|z_i),\\ z_i \sim q(.|X,A)}} \left[\log\Big(\frac{p(c_i|z_i)}{q(c_i|z_i)}\Big)\right]\\
& + N\sum_{i=1}^{N} \mathbb{E}_{\substack{c_j \sim q(.|z_j),\\ z_j \sim q(.|X,A)}} \left[\log\Big(\frac{p(c_j|z_j)}{q(c_j|z_j)}\Big)\right],\\
\geqslant & \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j \sim q(\gamma(.)|X,A)} \left[\log\Big(p(a_{ij}|z_i,z_j)\Big)\right] + 2N\sum_{i=1}^{N} \mathbb{E}_{z_i \sim q(\gamma(.)|X,A)} \left[\log\Big(\frac{p(z_i)}{q(\gamma(z_i)|X,A)}\Big)\right]\\
& + 2N\sum_{i=1}^{N} \mathbb{E}_{\substack{c_i \sim q(.|z_i),\\ z_i \sim q(.|X,A)}} \left[\log\Big(\frac{p(c_i|z_i)}{q(c_i|z_i)}\Big)\right],\\
\geqslant & \sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j \sim q(\gamma(.)|X,A)} \left[\log\Big(p(a_{ij}|z_i,z_j)\Big)\right] - 2N\sum_{i=1}^{N} KL\Big(q(\gamma(z_i)|X,A)||p(z_i)\Big)\\
& - 2N\sum_{i=1}^{N} \mathbb{E}_{z_i \sim q(.|X,A)} \left[KL\Big(q(c_i|z_i)||p(c_i|z_i)\Big)\right],\\
\geqslant & \mathcal{L}_4.
\end{aligned}
$$

The first term of $\mathcal{L}_4$ can be estimated based on Monte Carlo sampling (L^2 samples) and the reparameterization trick with a complexity $\mathcal{O}(dL^2N^2)$ as follows:

$$
\begin{aligned}
\sum_{i,j=1}^{N} \mathbb{E}_{z_i,z_j \sim q(\gamma(.)|X,A)} \left[\log\Big(p(a_{ij}|z_i,z_j)\Big)\right] \simeq & \ \frac{1}{L^2} \sum_{l_1,l_2=1}^{L} \sum_{i,j=1}^{N} \log\Big(p(a_{ij}|\gamma(z_i)^{(l_1)}, \gamma(z_j)^{(l_2)})\Big),\\
\simeq & \ \frac{1}{L^2} \sum_{l_1,l_2=1}^{L} \sum_{i,j=1}^{N} a_{ij} \log\Big(\text{Sigmoid}\Big((\gamma(z_i)^{(l_1)})^T \gamma(z_j)^{(l_2)}\Big)\Big)\\
+ & \ \frac{1}{L^2} \sum_{l_1,l_2=1}^{L} \sum_{i,j=1}^{N} (1-a_{ij}) \log\Big(1 - \text{Sigmoid}\Big((\gamma(z_i)^{(l_1)})^T \gamma(z_j)^{(l_2)}\Big)\Big).
\end{aligned}
$$

The second term of $\mathcal{L}_4$ can be computed analytically with a complexity $\mathcal{O}(dN)$ as follows:

$$
2\,N \sum_{i=1}^{N} KL\Big(q(\gamma(z_i)|X,A)||p(z_i)\Big) = N \sum_{i=1}^{N}\sum_{j=1}^{d} \Big(\mu^2_{\gamma(z_i)}[j] + \sigma^2_{\gamma(z_i)}[j] - \log(\sigma^2_{\gamma(z_i)}[j]) - 1\Big).
$$

The third term of $\mathcal{L}_4$ can be estimated based on Monte Carlo sampling (L samples) and the reparameterization trick with a complexity $\mathcal{O}(LN_cN)$ as follows:

$$2\,N\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(z_i|X,A)}\Big[KL\Big(q(c_i|z_i)||p(c_i|z_i)\Big)\Big] = 2\,N\sum_{i=1}^{N}\mathbb{E}_{z_i\sim q(z_i|X,A)}\Big[\sum_{c_i} q(c_i|z_i)\log\Big(\frac{q(c_i|z_i)}{p(c_i|z_i)}\Big)\Big],$$
$$\simeq 2\,\frac{N}{L}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{c_i} q(c_i|z_i^{(l)})\log\Big(\frac{q(c_i|z_i^{(l)})}{p(c_i|z_i^{(l)})}\Big).$$

Appendix F: Proposed Algorithm

Algorithm Training of FT-VGAE.

Input: Features matrix: X, Adjacency matrix: A, Number of iterations of the first phase: T_1, Number of iterations of the second phase: T_2, Number of neighbors: N_m, Number of over-clusters: N_o, Number of clusters: N_c.
Output: Clustering assignment matrix: H.

```
// First phase
for $i = 0$ to $T_1$ do
    Compute $\mathcal{L}_1$ according to Eq. (8)
    Update $W$ to maximize $\mathcal{L}_1$ using Adam optimizer;
end for
// Second phase
MinID ← $+\infty$;
for $i = 0$ to $T_2$ do
    Compute $\mathcal{L}_2$ according to Eq. (10)
    Update $W$ to maximize $\mathcal{L}_2$ using Adam optimizer;
    Compute ID;
    if ID < MinID then
        MinID ← ID;
        Save($W$);                 ▷ save the training weights of the epoch with the lowest ID
    end if
end for
Initialize $\{\Omega_j\}_{j=1}^{N_o}$, and $\{\Phi_j\}_{j=1}^{N_c}$ using k-means;
Load the saved training weights $W$;
// Third phase
$i \leftarrow 0$;
Compute ID and LID;
while ID < LID do
    if $i\%2 == 0$ then
        Compute $\mathcal{L}_3$ according to Eq. (16)
        Update $W$, $\{\Omega_j\}_{j=1}^{N_o}$ to maximize $\mathcal{L}_3$ using Adam optimizer;
    else
        Compute $\mathcal{L}_4$ according to Eq. (20)
        Update $W$, $\{\Phi_j\}_{j=1}^{N_c}$ to maximize $\mathcal{L}_4$ using Adam optimizer;
    end if
    Compute ID, LID, and DBI;
    Save(i, $W$);              ▷ save the training weights of all the epochs. Epochs are indexed by $i$
    Save(i, ID, LID, DBI);     ▷ save the metrics ID, LID, and DBI of all the epochs. Epochs are indexed by $i$
    $i \leftarrow i + 1$;
end while
Load the training weights of the epoch with the lowest ID among the 10 epochs with the best DBI;
Compute $Z$;
Compute $H$ by applying spectral clustering to $Z$;
return H
```

Appendix G: Complexity analysis

In Table 1, we compute the time and space complexity of our model FT-VGAE for the different phases. The time complexity for computing the embedded representations using the GCN architecture is equal to $\mathcal{O}(PJ|\mathcal{E}| + PJ^2N)$ for the three phases. The time complexity for computing the loss functions of: (1) the first phase is equal to $\mathcal{O}(JN + JL^2N^2)$, (2) the second phase is equal to $\mathcal{O}(JN_mN + JN_mL^2N^2)$, and (3) the third phase is equal to $\mathcal{O}(2JN + L(N_o + N_c)N + 2JL^2N^2)$. Hence, we find that the time complexity for: (1) the first phase is equal to $\mathcal{O}(PJ|\mathcal{E}| + (PJ^2 + J)N + JL^2N^2) = \mathcal{O}(JL^2N^2)$, (2) the second phase is equal to $\mathcal{O}(PJ|\mathcal{E}| + (PJ^2 + JN_m)N + JN_mL^2N^2) = \mathcal{O}(JN_mL^2N^2)$, and (3) the third phase is equal to $\mathcal{O}(PJ|\mathcal{E}| + (PJ^2 + LN_o + LN_c + 2J)N + 2JL^2N^2) = \mathcal{O}(2JL^2N^2)$. The space complexity for storing the embeddings and the training weights of the GCN layers is equal to $\mathcal{O}(PJ^2 + PJN)$. The space complexity for storing the decoded representations of: (1) the first phase is equal to $\mathcal{O}(L^2N^2)$, (2) the second phase is equal to $\mathcal{O}(N_mL^2N^2)$, and (3) the third phase is equal to $\mathcal{O}(L^2N^2)$. Furthermore, the space complexity for storing the embedded centers and the clustering assignments for the third phase is equal to $\mathcal{O}((N_c + N_o)(N + J))$. Hence, the space complexity amounts to : (1) $\mathcal{O}(PJ^2 + PJN + L^2N^2) = \mathcal{O}(L^2N^2)$ for the first phase, (2) $\mathcal{O}(PJ^2 + PJN + N_mL^2N^2) = \mathcal{O}(N_mL^2N^2)$ for the second phase, and (3) $\mathcal{O}((N_c + N_o)J + PJ^2 + (PJ + N_c + N_o)N + L^2N^2) = \mathcal{O}(L^2N^2)$ for the third phase.

Complexity	**First phase**	**Second phase**	**Third phase**
Time complexity	$\mathcal{O}(JL^2N^2)$	$\mathcal{O}(JN_mL^2N^2)$	$\mathcal{O}(2JL^2N^2)$
Space complexity	$\mathcal{O}(L^2N^2)$	$\mathcal{O}(N_mL^2N^2)$	$\mathcal{O}(L^2N^2)$

Table 1: Time and space complexity of our algorithms for the different phases. First row stands for the time complexity of one single iteration and the second row stands for the memory complexity. For simplicity, we assume that the number of features for all layers is a constant, and we omit the memory for storing the input graph because it is the same for all phases. Furthermore, we assume that $N \gg |\mathcal{E}|$, N_o, N_c, P, J, and L. $|\mathcal{E}|$ is the number of edges in the input graph $\mathcal{G}$. P is the number of layers of our GCN. J is the number of features for each layer (including the input matrix X). L is the number of MC samples. N_o is the number of over-clusters. N_c is the number of clusters.

Appendix H: Hyper-parameter settings

Our model require five hyperparameters (L, T_1, T_2, N_m, and N_o) without considering the architecture and the optimizer specifications. Three among these parameters are fixed for all considered datasets. Particularly, we adopt a Monte Carlo estimation based on a single sample ($L = 1$) similar to previous variational graph auto-encoders [Kipf and Welling, 2016], [Hui *et al.*, 2020]. We train for 200 iterations for the first phase similar to VGAE [Kipf and Welling, 2016] and GMM-VGAE [Hui *et al.*, 2020]. Furthermore, we find that fixing T_2 to 100 is sufficient for the second phase. The number of clusters N_c is set equal to the number of classes for each dataset.

In Table 2, we specify the hyper-parameters that we keep fixed for the six datasets. Specifically, we maintain the same architecture, the same number of training iterations T_1 and T_2, the same number of Monte Carlo samples L, the same optimizer, and the same learning rates for all datasets. In Table 3, we specify the two remaining hyper-parameters N_m and N_o for each dataset. N_m and N_o are selected from the ranges $[3, 5, 7, 9, 11]$ and $[50, 100, 150, 200, 250]$, respectively, using grid search.

Table 2: Common settings of FT-VGAE for the tested datasets.

Parameter	**Value**
Dimension of the first GCN layer	32
Dimension of the second GCN layer	16
Number of iterations of the first phase: T_1	200
Number of iterations of the second phase: T_2	100
Number of Monte Carlo samples: L	1
Optimizer	Adam
Learning rate of the first phase	0.01
Learning rate of the second phase	0.001
Learning rate of the third phase	0.001

Table 3: Data-specific settings of FT-VGAE for the tested datasets.

Parameter	**Cora**	**Citeseer**	**Pubmed**	**Brazil Air-traffic**	**Europe Air-traffic**	**US Air-traffic**
Number of neighbors: N_m	3	3	3	2	3	2
Number of over-clusters: N_o	40	150	20	10	20	10

Appendix I: Data description and preprocessing

In Table 4, we summarize the data statistics for the different datasets. Our empirical evaluation includes three citation networks [Sen *et al.*, 2008] (Cora, Citeseer, and Pubmed) and three air traffic graphs [Ribeiro *et al.*, 2017] (Brazil Air Traffic, Europe Air Traffic, and US Air Traffic). The nodes of an air traffic graph correspond to the airports and the graph edges capture the existence of commercial $f$lights between the airports. The labels associated with the airports indicate the level of activity. The level of activity of each airport is estimated based on the quartiles of the empirical distribution characterized by the total number of landings plus takeoffs of each airport. The nodes of the air traffic datasets do not have attributes. Similar to [Wu *et al.*, 2019], we construct the feature matrix with the one-hot encoding of node degrees. The nodes of the citation networks correspond to scientific publications and the graph edges capture the citations. For all datasets, the feature matrix X is (row-)normalized with the second norm $\|.\|_2$.

Table 4: Dataset statistics. $-$ indicates that the dataset comes without attributes.

Dataset	**Cora**	**Citeseer**	**Pubmed**	**Brazil Air Traffic**	**Europe Air Traffic**	**US Air Traffic**
Number of nodes	2708	3327	19717	131	399	1190
Number of edges	5429	4732	44338	1038	5995	13599
Number of features	1433	3703	500	$-$	$-$	$-$
Number of classes	7	6	3	4	4	4

Appendix J: Hardware and software configurations

All experiments are performed on a Linux server under the same hardware and software environments. The specification of the software libraries and frameworks as well as the hardware is provided in Table 5.

Table 5: Hardware and software used for all the conducted experiments.

Hardware	
RAM	132 GB
CPU model	Intel(R) Xeon(R) CPU E5-2620 V4 @ 2.10GHz
Number of CPUs	32
GPU model	GeForce RTX 2080 Ti
GPU memory	11 GB
Number of GPUs	2
Software	
Operating System	Ubuntu 18.04.5 LTS
Python	3.8.8
PyTorch	1.7.0
Sklearn	0.24.1

Appendix K: Intrinsic Dimension and Linear Intrinsic Dimension

ID (Intrinsic Dimension) and LID (Linear Intrinsic Dimension) assess the geometric transformation during the training process. The first metric (i.e., ID) describes the minimum number of parameters required to precisely capture the principal features. We estimate the ID of the latent manifolds based on TwoNN [Facco *et al.*, 2017]; a recent estimator that only considers the two nearest neighbors of each sample. TwoNN is computationally efficient and does not require the data density to effectively estimate the ID of highly-curved and non-uniformly sampled manifolds.

Let $X = \{x_i\}_{i=1}^N$ be a set of N points uniformly sampled from a data manifold, whose intrinsic dimension is equal to d. $r_1(i)$ and $r_2(i)$ are the distances between the sample x_i and its first and second nearest neighbors, respectively, among the set X. Let μ_i be the ratio between $r_2(i)$ and $r_1(i)$ (i.e., $\mu_i = r_2(i)/r_1(i)$). If the density between each point x_i and its second neighbor is constant, it has been proved [Facco *et al.*, 2017] that the ratio of a sample μ_i follows the Pareto distribution with a scale parameter equal to 1 and a shape parameter equal to d. Let $f(.|d)$ be the probability density function and $F(.|d)$ the cumulative distribution function of this Pareto distribution such that:

$$f(\mu_i|d) = d\,\mu_i^{-(d+1)} 1_{[1,+\infty]}(\mu_i), \qquad F(\mu_i|d) = (1 - \mu_i^{-d})\, 1_{[1,+\infty]}(\mu_i). \tag{1}$$

By simple algebra, we can derive the intrinsic dimension d from $F(\mu_i|d)$ as follows:

$$d = \frac{log(1 - F(\mu_i))}{log(\mu_i)}. \tag{2}$$

The empirical cumulative distribution function of μ_i is $F^{emp}(\mu_{\sigma(i)}) = i/N$, where σ is a permutation function that arranges the different μ_i for all $i \in [|1, N|]$ in ascending order. Hence, we can estimate d with a linear regression on the dataset $\left\{\left(log(\mu_i), -log\big(1 - F^{emp}(\mu_i)\big)\right)\right\}_{i=1}^{N}$.

LID (Linear Intrinsic Dimension) represents the dimension of the best subspace (with minimal rank) enclosing the data manifold. To estimate LID, we can use PCA (Principal Component Analysis) to identify the principal components (eigenvectors of the data's covariance matrix) that spans the subspace with the minimal projection error similar to [Ansuini *et al.*, 2019]. The difference between LID and ID indicates to what extent the data manifold is curved. For a highly-curved manifold, the linear intrinsic dimension is largely higher than the real intrinsic dimension (LID $\gg$ ID). For a flat manifold, the linear intrinsic dimension is equal to the real intrinsic dimension (LID $\approx$ ID).

Appendix L: Time comparison

In Table 6, we compare the execution time of FT-VGAE and two other variational graph auto-encoder models (i.e., VGAE and GMM-VGAE). Furthermore, we inspect the training time of FT-VGAE in each phase. We run each experiment ten times and we report the best, mean, and variance in execution time. We observe that the training time of FT-VGAE is higher than the training time of VGAE and GMM-VGAE. As the results show, the second and third phases are behind this increase in run-time. In accordance with the complexity of each phase, the training time of FT-VGAE remains reasonable considering the gain in clustering results.

Table 6: Analysing the running time (in seconds) of our approach and comparing with VGAE, GMM-VGAE.

Method	Cora			Citeseer			Pubmed		
	Best	Mean	Variance	Best	Mean	Variance	Best	Mean	Variance
First phase	1.725	1.930	0.078	1.789	1.916	0.055	17.974	18.084	0.042
Second phase	22.174	22.598	0.042	22.610	22.856	0.066	1622.551	1691.810	1705.517
Third phase	23.287	23.714	0.038	23.094	23.5	0.038	697.219	715.36	113.237
VGAE	1.692	1.868	0.058	1.789	1.968	0.085	17.090	17.289	0.071
GMM-VGAE	5.958	6.168	0.055	5.567	6.045	0.139	34.803	35.191	0.277
FT-VGAE	47.436	48.241	0.349	47.775	48.273	0.257	2359.002	2425.256	1828.270

References

[Ansuini *et al.*, 2019] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *NeurIPS*, pages 6111–6122, 2019.

[Facco *et al.*, 2017] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7(1):1–8, 2017.

[Hui *et al.*, 2020] Binyuan Hui, Pengfei Zhu, and Qinghua Hu. Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning. In *AAAI*, volume 34, pages 4215–4222, 2020.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NeurIPS Workshop*, pages 1–3, 2016.

[Ribeiro *et al.*, 2017] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *KDD*, pages 385–394, 2017.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–93, 2008.

[Wu *et al.*, 2019] Jun Wu, Jingrui He, and Jiejun Xu. Net: Degree-specific graph neural networks for node and graph classification. In *KDD*, pages 406–415, 2019.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.