Rethinking Graph Auto-Encoder Models for Attributed Graph Clustering

Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, and Riadh Ksantini

Abstract—Most recent graph clustering methods have resorted to Graph Auto-Encoders (GAEs) to perform joint clustering and embedding learning. However, two critical issues have been overlooked. First, the accumulative error, inflicted by learning from noisy clustering assignments, degrades the effectiveness of the clustering model. This problem is called Feature Randomness. Second, reconstructing the adjacency matrix sets the model to learn irrelevant similarities for the clustering task. This problem is called Feature Drift. Furthermore, the theoretical relation between the aforementioned problems has not yet been investigated. We study these issues from two aspects: (1) there is a trade-off between Feature Randomness and Feature Drift when clustering and reconstruction are performed at the same level, and (2) the problem of Feature Drift is more pronounced for GAE models, compared with vanilla auto-encoder models. Thus, we reformulate the GAE-based clustering methodology. Our solution is two-fold. First, we propose a sampling operator that triggers a protection mechanism against Feature Randomness. Second, we propose an operator that triggers a correction mechanism against Feature Drift by gradually transforming the reconstructed graph into a clustering-oriented one. As principal advantages, our solution grants a considerable improvement in clustering effectiveness and can be easily tailored to GAE models.

Index Terms—Unsupervised learning, graph clustering, graph auto-encoders

1 INTRODUCTION

Most recent attributed graph clustering methods leverage graph embedding [1], [2]. This strategy consists of projecting the graph structure and the node content in a low-dimensional compact space to harness the complementary modalities of attributed graphs. Graph embedding usually achieves exploitable representations for the clustering task [3]. A significant part of the graph embedding literature revolves around edge modeling [4], matrix factorization [5], and random walks [6]. Yet, these methods fall short of the expressive power of deep learning.

The last years witnessed the emergence of a promising graph embedding strategy, referred to as Graph Neural Networks (GNNs) [7]. GNNs extend the deep learning framework to graph-structured data. Among the prominent categories of GNNs, we find Graph Convolutional Networks (GCNs) [8], which generalize the convolution operation to graph data. Specifically, the intuition of the graph convolutional operation is to exploit the graph structure by smoothing the content features of each node over its neighborhood. Motivated by GCNs, Graph Auto-Encoders (GAEs) [9] and Variational Graph Auto-Encoders (VGAEs) [9] have shown notable achievements in several attributed graph clustering applications [10], [11], [12]. Typical GAE-based clustering methods project the input data in a low-dimensional space using graph convolutional layers and then reconstruct the adjacency matrix. Minimizing the reconstruction objective for the clustering task rules out the situation where the encoder is only trained based on noisy clustering assignments. The accumulated error makes the trained model capture non-representative features [13], which in turn corrupt the latent structure of the data. We adopt the terminology of Feature Randomness (FR) from our previous work [14] for investigating this problem in the context of GAEs.

As mentioned before, adding the decoder component is key to optimizing the reconstruction objective, which is a handy way to lower FR’s effect. However, the nature of the reconstructed graph is generally problematic to the clustering task. First, real-world graphs carry noisy and clustering-irrelevant links that can mislead the model into grouping together nodes from different clusters. This aspect can cause an under-segmentation problem. Second, it is also common for real-world graphs to come in a highly sparse structure. As a result, poor connectivity within the same cluster gives rise to an over-segmentation problem. Besides, the controversial relationship between clustering and reconstruction makes it hard to identify a static balance between them during the training process. This problem, which is referred to as Feature Drift (FD) in our previous work [14], remains unexplored for GNNs.

To address the aforementioned issues, we reformulate the GAE-based clustering methodology from an FR and FD perspective. We start by organizing the existing approaches...
into two groups, and we provide abstract formulations for each one. Next, we leverage the abstract description to examine the limitations of existing methods. Then, we provide formal characterizations of problems associated with the analyzed formulations on the authority of recent insights. After that, we propose a new conceptual design, which can mitigate the impact of FR and FD.

To put our conceptual design into action, we propose two operators, which can be easily integrated into GAE-based clustering methods. Possible options for addressing FR are: (1) operationalizing a correction mechanism that can reverse the randomness effect, (2) supplying the model with a protection mechanism that can exclude the sources of randomness as much as possible. Recently, the authors of [15], [16] have observed that pretraining a network with random labels then fine-tuning with clean ones leads to considerably lower accuracy, compared to a network trained with clean labels from the beginning. From this standpoint, we advocate accounting for FR using a protection strategy. Specifically, we design a sampling operator, prioritizing correctness by considering the difference between the first high-confidence and second high-confidence clustering assignment scores.

Additionally, we conceive a second operator that can control the effect of FD. Our design capitalizes on converting a general-purpose objective function into a task-specific one. Unlike previous GAE-based approaches, which optimize static objective functions during the whole clustering process, we gradually eliminate the graph reconstruction cost in favor of a clustering-oriented graph construction objective. Furthermore, our second operator contributes to preventing the over-segmentation and under-segmentation problems. More specifically, we gradually update the self-supervision graph by adding clustering-friendly edges and dropping clustering-irrelevant links.

The algorithmic intuitions behind our conceptual design and operators are supported by theoretical and empirical results. Theoretically, we demonstrate the existence of a trade-off between FR and FD for GAE-based clustering. Under mild assumptions, we prove that the graph convolutional operation and performing clustering and reconstruction at the same level aggravate the FD problem. Experimentally, we show that our operators can significantly improve the clustering effectiveness of existing GAE models without causing runtime overheads. Moreover, we show that our operators can mitigate the impact of FR and FD, and we provide empirical evidence that the improvement is imputed to the capacity of our operators in handling the trade-off between FR and FD. The significance of this work can be summarized as follows:

- Analysis: We organize GAE-based clustering approaches into two groups, and we provide abstract formulations for each one. Accordingly, we analyze and formalize the problems associated with the examined formulations. Then, we present a new conceptual design that can favorably control the trade-off between FR and FD. From a theoretical standpoint, we prove the existence of this trade-off, and we study two important aspects that differentiate GAE models from vanilla auto-encoder methods. Specifically, we investigate the impact of performing clustering and reconstruction at different layers on FR and FD. Moreover, we inspect the influence of the graph convolutional operation on FD.

- Methods: First, we propose a sampling operator \( \Xi \) that triggers a protection mechanism against FR. Second, we propose an operator \( \Upsilon \) that triggers a correction mechanism against FD by gradually transforming the reconstructed graph into a clustering-oriented one.

- Experiments: We conduct extensive experiments to investigate the behavior and profit from using our operators. Our empirical results provide strong evidence that the proposed operators improve the clustering performance of GAE models in effectiveness, by mitigating the effect of FR and FD.

2 \hspace{1cm} A NEW VISION FOR GAE-BASED CLUSTERING

This section advocates a new vision for building GAE-based clustering models beyond the classical perception of designing better clustering objectives. We begin by describing the existing GAE methods, which we organize into two groups. While the first group contains models that separate clustering from embedding learning, the second group only considers methods that perform joint clustering and embedding learning. For each group, we devise abstract formulations, and we study their associated limitations. Finally, we propose a new conceptual design to mitigate the examined problems. We consider our work to be the first initiative to analyze GAE-based clustering models from FR and FD perspectives.

2.1 Definitions and Notation

For a matrix \( M \), the expression \( M[i,:] \) denotes the \( i \)th row, and \( M[:,j] \) denotes the \( j \)th column of this matrix. For a set \( S \), the notation \( M[s,:] \) denotes a sub-matrix of \( M \) that has its rows indexed by the set \( s \). If \( \mathcal{F} \) is a function, then \( \mathcal{F}(X) \) denotes the output of the function \( \mathcal{F} \) after taking \( X \) as input. If \( X \) and \( Y \) are variables (i.e., scalars, vectors, matrices, tensors), then \( X(Y) \) denotes the order of dependency between \( X \) and \( Y \).

We consider a non-directed attributed graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, X) \), where \( \mathcal{V} = \{v_1, v_2, ..., v_N\} \) is a set of nodes with \( |\mathcal{V}| = N \), and \( N \) is the number of nodes. \( \mathcal{E} = \{e_{ij}\} \) represents the set of edges. The topological structure of the graph \( \mathcal{G} \) is denoted by the adjacency matrix \( A = (a_{ij}) \in \mathbb{R}^{N \times N} \), where \( a_{ij} = 1 \) if \((v_i, v_j) \in \mathcal{E}\) and \( a_{ij} = 0 \) otherwise. \( X = \{x_1, ..., x_N\} \) represents the matrix of features, where \( x_i \in \mathbb{R}^d \) is the feature vector associated with the node \( v_i \), and \( J \) is the dimensionality of the input space. We consider that the graph \( \mathcal{G} \) can be clustered into \( K \) clusters \( \{C_k^{clus}\}_{k=1}^K \).

Our study investigates the auto-encoding architecture for attributed graph clustering. Consequently, two functions should be specified. The first one is a non-linear encoder, which takes as inputs \( X \) and \( A \), and outputs low-dimensional latent representations denoted by the matrix \( Z \in \mathbb{R}^{N \times d} \); \( d \) denotes the dimension of the latent space. The set of learnable weights is denoted by \( \theta \). The second function is a decoder, which outputs a matrix \( \hat{A} = \text{sigmoid}(ZZ^T) \); \( \hat{A} \) measures the pairwise similarities between the latent codes.

Pseudo-supervision identifies the semantic categories of the data by constructing pseudo-labels and training the model with them. These pseudo-labels are generally predicted using

---

*IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 35, NO. 9, SEPTEMBER 2023*
a clustering algorithm. It then follows that part of the pseudo-labels are not aligned with the ground-truth labels.

We consider that the nodes of $\mathcal{G}$ are associated with $K$ ground-truth labels defining $K$ target clusters $\{C_{k}^{\text{clus}}\}_{k=1}^{K}$. We introduce two versions of the initial graph $\mathcal{G}$. A clustering graph $\mathcal{G}^{\text{clus}}$ defined by the adjacency matrix $A^{\text{clus}} = (a_{ij}^{\text{clus}}) \in \mathbb{R}^{N \times N}$, and a supervision graph $\mathcal{G}^{\text{sup}}$ defined by the adjacency matrix $A^{\text{sup}} = (a_{ij}^{\text{sup}}) \in \mathbb{R}^{N \times N}$. The adjacency matrices of the clustering and supervision graphs are expressed as follows:

$$
a_{ij}^{\text{clus}} = \begin{cases} 
\frac{1}{|C_{k}^{\text{clus}}|} \text{ if } \exists k \text{ such that } i, j \in C_{k}^{\text{clus}} \\
0 \text{ otherwise,} 
\end{cases}
$$

$$
a_{ij}^{\text{sup}} = \begin{cases} 
\frac{1}{|C_{k}^{\text{sup}}|} \text{ if } \exists k \text{ such that } i, j \in C_{k}^{\text{sup}} \\
0 \text{ otherwise.}
\end{cases}
$$

The idea of self-supervision involves solving a pretext task that requires a high-level understanding of the data. Specifically, the reconstruction loss is among the standard self-supervision methods for pretraining GAE models. It is generally expressed as a binary cross-entropy $L_{\text{bce}}(A(Z(\theta)), A^{\text{self}})$, where $A^{\text{self}}$ is the adjacency matrix of a self-supervision graph $\mathcal{G}^{\text{self}}$, and is set equal to $A$. Let $\Delta_C$ be a clustering algorithm and $P \in \mathbb{R}^{N \times K}$ be the clustering assignment matrix obtained by applying $\Delta_C$ to latent representations. $L_{\text{clus}}(P(Z(\theta)))$ is the clustering loss associated with algorithm $\Delta_C$. Without a pretraining stage, the clustering algorithm would be applied to random latent representations.

### 2.2 Formulations of the Deep Graph Clustering Methods

As mentioned in the beginning of this section, we organize existing approaches into two groups. For the first group, clustering is performed separately from embedding learning. Thus, we express the formulation associated with models from the first group as:

$$
P^{*} = \arg \min_{P} L_{\text{clus}}(P(Z(\theta))),
$$

where $\theta$ is initialized by the pretraining task, and $P^{*}$ is a solution to Eq. (1). Examples from the first group include MGAE (Marginalized Graph Auto-Encoder) [17], which improves clustering performance by increasing robustness to small input disturbances. From the same group, ARGAE (Adversarially Regularized Graph Auto-Encoder) [10] leverages an adversarial regularization technique that enforces embeddings to match a prior distribution using a discriminator network. Nevertheless, methods from the first group, such as MGAE and ARGAE, lack the capacity to learn clustering-oriented features.

In another perspective, Ansuini et al. [18] have shown that the latent codes of a deep network lie in curved manifolds highly. This aspect implies that the euclidean geometry is not suitable for assessing embedded similarities after the pretraining phase. To alleviate this problem, the second group of GAE-based clustering methods achieves joint clustering and embedded learning. In this regard, we reformulate Eq. (1) in a way that forces the embedded representations to follow the clustering assumptions. To ensure this quality, the formulation of the second group is articulated as:

$$
\theta^{*}, P^{*} = \arg \min_{\theta, P} L_{\text{clus}}(P(Z(\theta))),
$$

where $\theta^{*}$ and $P^{*}$ are solutions to Eq. (2). Typical clustering losses aim at decreasing the intra-cluster variance and increasing the inter-cluster variance. By optimizing $\theta$, the embedded points move in a way that establishes a clustering-oriented distribution. Therefore, the choice of the clustering cost becomes less important. However, the formulation of Eq. (2) is still problematic because the embedded points can move in a way that violates their semantic categories, while still decreasing the embedded clustering penalty.

Let $Q$ be the matrix of true hard clustering assignments. A supervised deep clustering problem can be described by Eq. (3). Compared with Eq. (2), the supervised objective pushes the latent codes to be clustering-friendly according to the true clustering assignment matrix $Q$. Let $\Lambda_{H}$ be the Hungarian algorithm [19], which finds the best linear mapping from a true clustering assignment matrix $Q$ to a predicted clustering assignment matrix $P$. The algorithm $\Lambda_{H}$ outputs a matrix $Q^{*} = \Lambda_{H}(Q, P)$. By analogy with pseudo-supervision, $y(P) = \arg \max_{y \in \{0, \ldots, K\}} (P_{y})_{j}$ can be considered as pseudo-labels for solving Eq. (2), and $y(Q) = \arg \max_{y \in \{0, \ldots, K\}} (Q_{y})_{j}$ can be considered as ground-truth labels for solving Eq. (3). The ultimate goal of deep clustering is to formulate an optimization problem, where a solution for the clustering assignment matrix is $P^{*}$, such that $y(P^{*}) = y(Q^{*})$.

$$
\theta^{*} = \arg \min_{\theta} L_{\text{clus}}(Q(Z(\theta)))).
$$

**Feature Randomness:** Under the extreme condition of entirely random labels, Zhang et al. [20] have shown that an over-parameterized neural network can perfectly fit the training set. This finding inspired the scientific community to investigate the difference between “training with true labels” and “training with random labels”. Keskar et al. [21] have proposed a metric to measure the sharpness of a minimizer to assess generalization. In [18], [22], the authors have investigated the intrinsic dimensionality of the embedded representations to understand the impact of random labels. However, the impact of random labels on Graph Neural Networks and graph datasets remains unexplored.

Feature Randomness occurs when a neural network is trained using pseudo-labels. In this context, the network can learn random features that capture irrelevant similarities; features that retain little to no correlation between the training samples and their corresponding true labels. We can characterize this problem by the gradient deviation caused by introducing pseudo-labels. In our previous work [14], we have proposed a metric $\Lambda_{FR}$ that assesses the effect of FR. The $\Lambda_{FR}$ metric measures the cosine similarity between the gradient of the supervised loss and the gradient of the pseudo-supervised loss as described in Eq. (4). $\Lambda_{FR}$ lies within the range $[-1, 1]$. Higher values are associated with lower FR.
\[ \Lambda_{FR} = \cos \left( \frac{\partial L_{clus}(P(Z(\theta)))}{\partial \theta}, \frac{\partial L_{clus}(Q(Z(\theta)))}{\partial \theta} \right). \]  

ACC is among the standard metrics for assessing the clustering process. It measures the similarity between \(y(P)\) and \(y(Q)\). Unlike ACC, \(\Lambda_{FR}\) computes the cosine similarity between the gradient vectors associated with \(y(P)\) and \(y(Q)\). The evolution of the gradient vectors grants a better understanding of the optimization space topology. From this perspective, we advocate the use of gradient-based metrics to identify and investigate problems related to gradient-based optimization.

Possible strategies for countering random projections are: (1) performing pseudo-supervision based on self-paced training, and (2) pretraining by self-supervision (pretext task) and finetuning by combining pseudo-supervision (main task) and self-supervision (pretext task). An example of the first strategy is AGE [23], which constructs a pseudo-supervised graph by linking pairs of high similarity and disconnecting the low similarity ones. However, two limitations are associated with the first strategy. First, it does not involve pretraining using a pretext task, and the pseudo-labels are initially constructed from the input data. Hence, the first strategy is limited to datasets, where the node features have low semantic similarities that can be extracted without neural networks. Second, the first strategy does not combine pseudo-supervision and self-supervision during the clustering phase. You et al. [24] have shown that combining the main task with a self-supervision pretext brings greater generalizability and robustness to GCNs. For the second strategy, adjacency reconstruction constitutes the standard self-supervised technique for GAE models.

In this work, we focus on the relation between pseudo-supervision (i.e., main task) and self-supervision (i.e., pretext task), which is governed by FR and FD. Accordingly, we reformulate Eq. (2) to take into consideration the reconstruction loss:

\[ \theta^*, P^* = \arg \min_{\theta, P} L_{clus}(P(Z(\theta))) + \gamma L_{bce}(\hat{A}(Z(\theta)), A^{self}), \]

where \(\theta^*\) and \(P^*\) are solutions to this equation. \(\gamma\) is a balancing hyper-parameter that controls the trade-off between clustering and reconstruction. Examples of this category include DAEGC (Deep Attentional Embedded Graph Clustering) [12], which employs an attention mechanism to adjust the influence of neighboring nodes. Another example is GMM-VGAE (Variational Graph Auto-Encoder with Gaussian Mixture Models) [11], which harnesses Gaussian Mixture Models to capture the variance between the different clusters. However, the strong competition between embedded clustering and reconstruction causes FD.

**Feature Drift:** As previously mentioned, some recent GAE-based clustering methods rely on a linear combination of embedded clustering and adjacency reconstruction. However, there is a strong trade-off between the two objectives. On the one hand, clustering aims at decreasing intra-cluster variance and increasing inter-cluster variance. On the other hand, the reconstruction objective pushes the latent representations to maintain all variances (i.e., intra-cluster and inter-cluster variances). The features learned by embedded clustering can be destroyed by the reconstruction cost, which captures clustering-irrelevant similarities.

Multi-objective optimization requires taking into account the trade-off between multiple objectives. Conflicting objective functions are those for which there is no solution that optimizes all of them simultaneously. In this case, multiple Pareto optimal solutions are possible. To select a single solution, additional subjective information is required. The validity of all possible solutions is the same without this subjective information.

Feature Drift occurs when two strongly competing loss functions are minimized simultaneously. The impact of FD can be estimated by measuring the level of competition between the two functions. In our previous work [14], we evaluate the level of competition using the cosine similarity between the gradient of the self-supervised loss and the gradient of the pseudo-supervised loss. This metric was only tested when clustering and reconstruction are computed based on euclidean distances. However, in the graph case, we found that this metric does not reveal any interpretive pattern, probably because the two loss functions have completely different formulations. Thus, we propose a new metric that measures the cosine similarity between two gradients of a similar formulation. The two gradients are computed based on the same loss formulation \(L_{bce}\) but with different configurations: the gradient of \(L_{bce}(A, A^{self})\) and the gradient of its supervised version \(L_{bce}(A, A^{sup})\). Our new metric is described by:

\[ \Lambda_{FD} = \cos \left( \frac{\partial L_{bce}(A, A^{self})}{\partial \theta}, \frac{\partial L_{bce}(A, A^{sup})}{\partial \theta} \right). \]
DynAE and DeepCluster) perform hard clustering using K-means and do not consider the covariance of the embedded clusters. Enforcing pseudo-labels obtained by a hard clustering algorithm may destroy relevant similarities and hence give rise to FR. Last but not least, none of these methods can take both the structural and content information as input signals.

2.3 The Proposed Conceptual Design

To overcome the limitations of previous methods, we propose a new conceptual design. Our solution fixes the deficiency of existing GAE models from the perspective of FR and FD. In Fig. 1, we illustrate the generic framework of this conceptual design. More precisely, our formulation depends on two operators. First, we develop a sampling operator \( \Xi \) to gradually spot nodes with reliable clustering assignments, denoted by the set \( \Omega \). We exploit the set of reliable nodes to optimize the embedded clustering objective. Second, we propose a graph-specific operator \( \Upsilon \) to gradually transform the general-purpose self-supervision signal \( A \) into a clustering-oriented self-supervision signal \( A_{\text{clus}} \).

The formulation of our conceptual design is expressed by:

\[
\theta^*, P^* = \underset{\theta, P}{\arg \min } \ L_{\text{clus}}(P(\Xi(Z(\theta)))) + \gamma L_{\text{hea}}(A(Z(\theta)), \Upsilon(A, P(\Xi(Z(\theta))), \Omega)).
\]

Clustering is performed on all samples to obtain the different assignments. The clustering assignments of the reliable nodes are denoted by \( P(\Xi(Z(\theta))) \). The clustering loss is only calculated on the set of reliable nodes as described by the notation \( L_{\text{clus}}(P(\Xi(Z(\theta)))) \). The operation \( \Upsilon(A, P(\Xi(Z(\theta))), \Omega) \) makes a single-step modification to the input graph \( A \). It only affects the sub-graph defined by the set reliable nodes \( \Omega \). As opposed to that, \( \Upsilon(A, Q'(Z(\theta)), \mathcal{V}) \) outputs the target clustering-oriented graph that we want to obtain at the end of the training process. It is generated by transforming the whole graph using the supervision signal.

3 Theoretical Analysis

Our conceptual design aims at reducing the impact of FD without causing excessive FR. An intuitive explanation of the trade-off between FR and FD is provided in the previous section. In this section, we discuss the problems of FR and FD for GAE models from a theoretical standpoint. Our formal analysis includes three points: (1) proving the existence of a trade-off between FR and FD for GAE models, (2) understanding the impact of performing clustering and reconstruction at different layers on FR and FD, and (3) understanding the impact of the graph convolutional operation, which is performed by all encoding layers, on FD. All mathematical proofs and derivations are described in the Appendices.

3.1 Trade-Off Between FR and FD

Let \( L_{c} \) be a generic loss, which takes as input an adjacency matrix \( A' = (a'_{ij}) \in \mathbb{R}^{N \times N} \) and a feature matrix \( Z' \in \mathbb{R}^{N \times d} \), and can be written in the form:

\[
L_{c}(Z', A') = \frac{1}{2} \sum_{1 \leq i, j \leq N} a'_{ij} \| z'_i - z'_j \|^2_2.
\]

Proposition 1. The reconstruction loss for a GAE model can be expressed as:

\[
L_{\text{bce}}(A(Z(\theta)), A_{\text{self}}) = L_{c}(Z(\theta), A_{\text{self}}) + L_{R}(Z(\theta), A_{\text{self}}),
\]

\[
L_{R}(Z(\theta), A_{\text{self}}) = \sum_{i,j} (\log(1 + \exp(z_i^T z_j)))
\]

\[
-\frac{1}{2} a_{ij} (\| z_i \|^2 + \| z_j \|^2).
\]

In Proposition 1, we write the reconstruction loss of a GAE model in the form of a linear combination between a graph Laplacian regularization term \( L_{c}(Z(\theta), A_{\text{self}}) \) and another loss \( L_{R}(Z(\theta), A_{\text{self}}) \). A trivial solution to minimize \( L_{c}(Z(\theta), A_{\text{self}}) \) consists of mapping the features of all nodes to the same latent code. State-of-the-art self-supervised methods rely on negative pairs [29], [30] or a cross-model supplementary loss function [31] to avoid degenerate solutions. In our case, the trivial solutions are ruled out by the function \( L_{R}(Z(\theta), A_{\text{self}}) \). More precisely, minimizing \( \log(1 + \exp(z_i^T z_j)) \) implies an increase in the angle between the two vectors \( z_i \) and \( z_j \) or a decrease in their norms if their angle is lower than 90° or an increase in their norms if their angle is greater than 90°. However, minimizing the second part of \( L_{R} \) (i.e., \( -\frac{1}{2} a_{ij} (\| z_i \|^2 + \| z_j \|^2) \)) increases the norm of \( z_i \) and \( z_j \) when there is a link between the two nodes \( v_i \) and \( v_j \). Hence, we can conclude that \( L_{R} \) increases the angle between each couple of vectors \( z_i \) and \( z_j \) if there is a link between them. Otherwise, decreasing the norm of both vectors might be sufficient.

Proposition 2. The k-means clustering loss applied to the embedded representations can be expressed as:

\[
L_{\text{clus}}(Z(\theta)) = L_{c}(Z(\theta), A_{\text{clus}}).
\]

In Proposition 2, we write the embedded k-means loss in the form of a graph Laplacian regularization loss \( L_{c}(Z(\theta), A_{\text{clus}}) \). As we can see, the graph required for embedded k-means is different from the graph required for the reconstruction loss. Furthermore, training the encoder to minimize embedded k-means without a reconstruction loss can easily lead to degenerate solutions.

Theorem 1. The linear combination between reconstruction and embedded k-means for a GAE model can be expressed as:

\[
\n\]
\[ L_{clus}(Z(\theta)) + \gamma L_{bce}(\hat{A}(Z(\theta)), \mathcal{A}^{self}) = \]
\[ L_{c}(Z(\theta), \mathcal{A}^{clus} + \gamma \mathcal{A}^{self}) + \gamma L_{R}(Z(\theta), \mathcal{A}^{self}). \]

In Theorem 1, we have a typical GAE-based clustering model that optimizes a linear combination between embedded k-means and reconstruction. Based on Proposition 1 and Proposition 2, we can write the loss function of this GAE model in the form of a linear combination between a graph-weighted loss \( L_{c}(Z(\theta), \mathcal{A}^{clus} + \gamma \mathcal{A}^{self}) \) and a regularization term \( L_{R}(Z(\theta), \mathcal{A}^{self}) \). The regularization term \( L_{R} \) enables the training process to avoid degenerate solutions. The graph associated with \( L_{c} \) is a combination between the clustering graph and the self-supervision graph. Based on this result, we can clearly spot the trade-off between FR and FD, which is caused by combining two graphs of different nature. These two graphs suffer from different problems. While the clustering graph has several inaccurate edges, the input graph is sparse and comes with clustering-irrelevant links. On the one hand, decreasing the balancing hyperparameter \( \gamma \) reinforces the impact of the clustering graph on the optimization process, which in turn gives rise to FR. On the other hand, increasing \( \gamma \) leads to higher levels of FD due to the high sparsity and clustering-irrelevant links of the self-supervision graph.

### 3.2 Impact of Performing Clustering and Reconstruction at Different Layers on FR and FD

To understand the impact of a GAE model on FR and FD compared with a vanilla auto-encoder model, we analyze \( \Lambda_{FR} \) and \( \Lambda_{FD} \) in a variety of contexts. To this end, we start by computing the gradient of the clustering and reconstruction losses w.r.t. the embedded representations.

**Proposition 3.** The gradient of the reconstruction loss \( L_{bce}(\hat{A}(Z(\theta)), \mathcal{A}^{self}) \) w.r.t. the embedded representation \( z_{i} \) can be expressed as:
\[
\frac{\partial L_{bce}(\hat{A}(Z(\theta)), \mathcal{A}^{self})}{\partial z_{i}} = \sum_{1 \leq j \leq N} (\hat{a}_{ij} - a^{self}_{ij}) z_{j}.
\]

**Proposition 4.** The gradient of the clustering loss \( L_{clus}(Z(\theta)) \) w.r.t. the embedded representation \( z_{i} \) can be expressed as:
\[
\frac{\partial L_{clus}(Z(\theta))}{\partial z_{i}} = \sum_{1 \leq j \leq N} a^{clus}_{ij} (z_{i} - z_{j}).
\]

In Proposition 3, we compute the gradient of the reconstruction loss, and in Proposition 4, we compute the gradient of the embedded k-means loss. To facilitate the theoretical analysis of FR and FD, we perform three simplifications. Since the trade-off between FR and FD is only related to the graph-weighted functions \( L_{c} \), we exclude the regularization term \( L_{R} \) from the gradient computation. Restraining our analysis to the \( L_{c} \) functions simplifies the analytical computation for evaluating FR and FD. In another simplification, we use the inner product for measuring the similarity between the gradient vectors instead of using the cosine function. Using the inner product overcomes the need to deal with the gradient norms. Furthermore, the cosine similarity can be seen as a special case of the inner product (inner product between normalized gradients). It is well-known that normalized gradients can improve the generalization capacity. If we assume that our training process does not lead to exploding or vanishing gradients, then the gradient norms have little impact on the inner product. Finally, the final simplification consists of using normalized graphs. We denote the normalized self-supervision adjacency matrix by \( \tilde{A}^{self} = D^{-\frac{1}{2}} A^{self} D^{-\frac{1}{2}} = (a^{self}_{ij})_{ij} \) where \( D = \text{diag}(d_{1}, ..., d_{n}) \) is the degree matrix of \( A^{self} \) such that \( d_{i} = \sum_{j=1}^{n} a^{self}_{ij} \). Furthermore, \( A^{clus} \) and \( A^{sup} \) are normalized matrices by definition. Based on the aforementioned simplifications, we can obtain elementary metrics for assessing FR and FD as explained by Definition 1 and Definition 2 respectively.

**Definition 1.** For a GAE model \( Q \), we define a metric \( \Lambda'_{FR}(Q, z_{i}) \) to evaluate the impact of FR at the level of an embedded point \( z_{i} \) as follows:
\[
\Lambda'_{FR}(Q, z_{i}) = \left\langle \frac{\partial L_{c}(Z(\theta), \mathcal{A}^{clus})}{\partial z_{i}} \left\| \frac{\partial L_{c}(Z(\theta), \mathcal{A}^{clus})}{\partial z_{i}} \right\|_{1} \right\rangle.
\]

**Definition 2.** For a GAE model \( Q \), we define a metric \( \Lambda'_{FD}(Q, z_{i}) \) to evaluate the impact of FD at the level of an embedded point \( z_{i} \) as follows:
\[
\Lambda'_{FD}(Q, z_{i}) = \left\langle \frac{\partial L_{c}(Z(\theta), \tilde{A}^{self})}{\partial z_{i}} \left\| \frac{\partial L_{c}(Z(\theta), \tilde{A}^{self})}{\partial z_{i}} \right\|_{1} \right\rangle.
\]

Modern neural networks are Lipschitz functions. The Lipschitz constant of a function informs how much the output can change in proportion to an input change. Constraining the Lipschitz constant of a neural network is connected to several interesting aspects. For instance, reducing this constant enhances adversarial robustness [32]. For classification, reducing the Lipschitz constant induces better generalization bounds as shown by several previous works [33], [34], [35]. In this section, we show the impact of constraining the Lipschitz constant on FR and FD for two specific situations. In the subsequent analysis, we use the following definition:

**Definition 3.** Given two metric spaces \((X, d_{X})\) and \((Y, d_{Y})\), where \( d_{X} \) is a metric on set \( X \) and \( d_{Y} \) is a metric on set \( Y \), a function \( f : X \to Y \) is called Lipschitz continuous if:
\[
\exists \tau_{1} \geq 0, \forall x_{1}, x_{2} \in X \| f(x_{2}) - f(x_{1}) \|_{d_{X}} \leq \tau_{1} \| x_{2} - x_{1} \|_{d_{X}},
\]
and the Lipschitz constant \( \tau_{1}^{*} \) of \( f \) is defined as:
\[
\tau_{1}^{*} = \sup_{x_{1}, x_{2} \in X} \left( \frac{\| f(x_{2}) - f(x_{1}) \|_{d_{Y}}}{\| x_{2} - x_{1} \|_{d_{X}}} \right).
\]
If \( f \) is a Lipschitz function and there exists \( \tau_{2} \geq 0 \) such that for all \( x_{1}, x_{2} \in X \| f(x_{2}) - f(x_{1}) \|_{d_{Y}} \geq \frac{\tau_{2}}{\| x_{2} - x_{1} \|_{d_{X}}}, \) then \( f \) is bi-Lipschitz. We denote the Lipschitz constant of \( f^{-1} : f(X) \to X \) as \( \tau_{2}^{*} \).
Unlike typical auto-encoder models for euclidean data clustering, GAE models perform clustering and reconstruction at the same level (i.e., same layer). We study the impact of performing clustering and reconstruction at different layers on FR and FD. To this end, we consider two possible scenarios. Let $\mathcal{N}(d, d', L)$ be a family of fully-connected layers denoted by $f$:

$$f : \mathbb{R}^d \to \mathbb{R}^{d'}, \quad z \mapsto \text{ReLU}(W_1z + b_1) + b_2,$$

such that $l = 1, \ldots, L$ indexes the different layers of the network $f$, $W_l \in \mathbb{R}^{d_l \times d_{l+1}}$, $b_l \in \mathbb{R}^{d_l}$, $d = d_0$, and $d' = d_L$. The first scenario consists of adding fully-connected encoding layers on top of the last graph convolutional layer, and performing clustering at the level of the last encoding layer. This scenario is illustrated in Fig. 2. The second scenario consists of adding fully-connected decoding layers on top of the last graph convolutional layer, and performing reconstruction at the level of the last decoding layer. This scenario is illustrated in Fig. 3. Accordingly, we compare the behaviour of a typical GAE-based clustering model with the two versions described by Figs. 2 and 3, in terms of FR and FD, at the level of the embedded representations.

**Theorem 2.** Given two GAE models $Q_1$ and $Q_2$, which have the same graph convolutional layers. $Q_1$ optimizes the objective function in Equation (7) and $Q_2$ minimizes the loss function in Equation (8), where $f \in \mathcal{N}(d, d', L)$ and $d' \ll d$. Let $\tau_1^f$ be the Lipschitz constant of $f$, $Z_i = (z_{ij} - z_{ij}')_{j \leq j_i} \in \mathbb{R}^{N \times d'}$, $\xi_i = (\|z_i - z_i\|_2)_{j \leq j_i} \in \mathbb{R}^N$, and $a_i$ is the $i$th row of $A$.

$$L_{Q_1} = L_{clus}(Z(\theta)) + \gamma L_{lbe}(\hat{A}(Z(\theta)), A^{self}),$$

$$L_{Q_2} = L_{clus}(f(Z(\theta))) + \gamma L_{lbe}(\hat{A}(f(Z(\theta))), A^{self}).$$  

\[ (7) \quad (8) \]

- $A'_{FD}(Q_2, z_i) = A'_{FD}(Q_1, z_i)$.
- If

$$\tau_1^f \leq \sqrt{\frac{(Z_i^T a_{sup})^T (Z_i^T a_{clus})}{(\xi_i^T a_{sup}) (\xi_i^T a_{clus})}},$$

then

$$A'_{FD}(Q_2, z_i) \leq A'_{FD}(Q_1, z_i).$$

In Theorem 2, we study the first scenario where a bunch of encoding layers is added on top of the last graph convolutional layer, and the clustering loss is applied at the level of the last encoding layer. We know that reducing the Lipchitz constant is linked to a better generalization capacity [35]. Based on Theorem 2, we found that a constrained Lipchitz constant of the network $f$ leads to more FR compared with the initial GAE-based clustering model. Furthermore, we found that FD is not affected by the added encoding layers. Hence, we conclude that adding encoding layers independently from the decoding operation increases FR without affecting FD. An intuitive interpretation of this result comes from the fact that the gradient of the reconstruction loss does not back-propagate through the added encoding layers. Therefore, the clustering loss becomes more prone to random projections.

**Theorem 3.** Given two GAE models $Q_1$ and $Q_2$, which have the same graph convolutional layers. $Q_1$ optimizes the objective function in Equation (9) and $Q_2$ minimizes the loss function in Equation (10), where $f \in \mathcal{N}(d, d', L)$ is an injective function and $d' \gg d$. Let $\tau_2^f$ be the Lipschitz constant of $f^{-1}$: $f(\mathbb{R}^d) \to \mathbb{R}^{d'}$, $Z_i = ((f(z_{ij}))_{j \leq j_i} - (f(z_{ij})))_{j \leq j_i} \in \mathbb{R}^{N \times d'}$, $\xi_i = (\|f(z_i) - f(z_i)\|_2)_{j \leq j_i} \in \mathbb{R}^N$, and $a_i$ is the $i$th row of $A$.

$$L_{Q_1} = L_{clus}(Z(\theta)) + \gamma L_{lbe}(\hat{A}(Z(\theta)), A^{self}),$$

$$L_{Q_2} = L_{clus}(Z(\theta)) + \gamma L_{lbe}(\hat{A}(f(Z(\theta))), A^{self}).$$  

\[ (9) \quad (10) \]

- $A'_{FD}(Q_2, z_i) = A'_{FD}(Q_1, z_i)$.
- If

$$\tau_2^f \leq \sqrt{\frac{(Z_i^T a_{sup})^T (Z_i^T a_{clus})}{(\xi_i^T a_{sup}) (\xi_i^T a_{clus})}},$$

then

$$A'_{FD}(Q_2, z_i) \leq A'_{FD}(Q_1, z_i).$$

In Theorem 3, we study the second scenario where a bunch of decoding layers is added on top of the last graph convolutional layer, and the reconstruction loss is applied at the level of the last decoding layer. This case is similar to the typical auto-encoder, where the decoder has several...
layers. Based on Theorem 3, we found that a constrained Lipchitz constant of $f^{-1}$ leads to less FD compared with the initial GAE-based clustering model. Intuitively, it is expected that the decoding layers attenuate the effect of FD when the gradient of the reconstruction loss has to back-propagate through several layers.

### 3.3 Impact of the Graph Convolutional Operation on FD

The graph convolutional operation constitutes a principal difference between a typical auto-encoder model and a GAE model. For this reason, we study the impact of this operation on the clustering task from the perspective of FD. Feature propagation for a single GCN layer is expressed by the rule $X^{(k+1)} = \phi(\hat{A}^{self} X^{(k)} W_k)$, where $X^{(k)}$ represents the node features of the $k$th layer, $W_k$ is the matrix of trainable weights associated with this layer, and $\phi$ is an activation function. The multiplication of the graph filter $\hat{A}^{self}$ with the graph signal $X^{(k)}$ defines the graph convolutional operation. Let $h$ be an aggregation function such that $h^{sup}(x_i) = \sum_j a_{ij}^{sup} x_j$ is the center of the true cluster associated with $x_i$ (computed based on ground-truth assignments), and $h^{self}(x_i) = \sum_j a_{ij}^{self} x_j$ is the center of the immediate neighbors of $x_i$ according to $A^{self}$. In Eq. (11), we define a function $\mathcal{P}$ to locally assess the impact of the graph filtering operation on the clustering task.

$$\mathcal{P}(x_i) = \|x_i - h^{sup}(x_i)\|_2 - \|h^{self}(x_i) - h^{sup}(x_i)\|_2. \quad (11)$$

If $\mathcal{P}(x_i) \geq 0$, we say that the graph filtering operation has a positive impact on clustering the node $v_i$. To understand the impact of the filtering operation on FD, we consider two possible scenarios.

**Assumption 1.** The self-supervision adjacency matrix $\hat{A}^{self}$ represents the immediate neighbors with a small error, that is,

$$\forall i, j \in [1, N], \text{ such that } \hat{a}_{ij}^{self} \neq 0, \quad x_i = x_j + e_{ij},$$

where $e_{ij} \in \mathbb{R}^d$ is a small error (i.e., $e_{ij}$ almost equal to zero).

**Assumption 2.** The immediate neighbors of a node $v_i$ are assumed to activate the same neurons for a well-trained ReLU-Affine layer with a training weight $W$, that is,

$$\forall i, j \text{ if } \hat{a}_{ij}^{self} \neq 0 \text{ then } \text{Sign}(W^T x_i) = \text{Sign}(W^T x_j).$$

**Theorem 4.** Given two models $Q_1$ and $Q_2$, which optimize the same objective function as described by Eq. (12). $Q_1$ has a single fully-connected encoding layer characterized by the function $f_1(X) = \text{ReLU}(X W)$, where $W$ represents the learning weights of this layer. $Q_2$ has a single graph convolutional layer characterized by the function $f_2(X) = \text{ReLU}(\hat{A}^{self} X W)$. If $\mathcal{P}(f_1(x_i)) \geq 0$ then $\Delta'_{FD}(Q_2, x_i) \leq \Delta'_{FD}(Q_1, x_i)$. In Theorem 4, we study the case, which consists of comparing a one-layer graph convolutional encoder against a one-layer fully connected encoder. Our proof depends on two reasonable properties of $\hat{A}^{self}$. Specifically, we know by definition that $\hat{A}^{self}$ connects each node with few immediate neighbors as opposed to $A^{sup}$, which connects each node with all nodes from the same true cluster. Assumption 1 states that the immediate neighbors of a node $v_i$ are represented with small errors. The second Assumption 2 asserts that the immediate neighbors of a node $v_i$ activate the same neurons for a well-trained layer. Under these mild assumptions, Theorem 4 indicates that performing a graph convolutional operation before a fully-connected layer increases the effect of FD on a node $v_i$ if the graph convolutional operation has a positive impact on clustering $v_i$. Intuitively, $\hat{A}^{self}$ only considers the immediate neighbors (due to the sparsity of $\hat{A}^{self}$) and maintains some clustering-irrelevant links. For every layer, we know that the graph convolutional operation is equivalent to minimizing the loss function $L_c(X^{(k)}, \hat{A}^{self})$ [8], which implies an increase of FD at the level of the same layer.

**Theorem 5.** Given two models $Q_1$ and $Q_2$, which optimize the same objective function as described by Eq. (13). $Q_1$ has a single graph convolutional layer characterized by the function $f_1(X) = \text{ReLU}(\hat{A}^{self} X W_{1i})$, where $W_1$ represents the learning weights of this layer. $Q_2$ has two graph convolutional layers characterized by the function $f_2(X) = \text{ReLU}(\hat{A}^{self} \text{ReLU}(\hat{A}^{self} X W_1) W_2)$, where $W_2$ represents the learning weights of the second layer. We suppose that the Lipschitz constant $\epsilon^2_f$ of the second graph convolutional layer is less than or equal to 1.

$$L_{Q_1} = L_{Q_2} = L_{clus}(Z(\theta)) + \gamma L_{bce}(\hat{A}(Z(\theta)), A^{self}). \quad (13)$$

**Under Assumption 1 and Assumption 2, we have:**

If $\mathcal{P}(f_1(x_i)) \geq 0$ then $\Delta'_{FD}(Q_2, x_i) \leq \Delta'_{FD}(Q_1, x_i)$.

In Theorem 5, we study the second scenario, which consists of comparing a one-layer graph convolutional encoder against a two-layer graph convolutional encoder. Similar to Theorem 4, our proof relies on Assumption 1 and Assumption 2. As a result, we found that adding a graph convolutional layer increases the effect of FD on a node $v_i$, if the graph convolutional operation has a positive impact on clustering $v_i$. Intuitively, the smoothing effect of each layer propagates to the embedded representations $Z$, which in turn drift the clustering-oriented structures. For instance, an infinite-depth graph convolutional network produces the same embedded vector for each node [36]. Mapping all nodes to the same embedded point renders the clustering irrelevant.

### 4 Proposed Operators

Our theoretical analysis indicates the limitations of GAE models in tackling the FR and FD problems. Motivated by these limitations, we propose two operators that can be easily integrated into existing models. Most importantly, our operators gradually transform the general-purpose self-supervision graph into a clustering-oriented graph. First, we design a sampling operator $\Xi$ that triggers a protection mechanism against FR. More precisely, $\Xi$ can delay FR from
quickly taking place. Second, we propose an operator \( \Upsilon \) that triggers a correction mechanism against FD. \( \Upsilon \) revokes the impact of FD by gradually transforming the reconstructed graph into a clustering-oriented one.

### 4.1 A Protection Mechanism Against FR

The process of collecting large-scale annotated data is generally associated with the introduction of corrupted labels (i.e., labels with incorrect class labels). Some supervised methods [37], [38], [39] handle the impact of corrupted labels using an iterative selection protocol. Specifically, samples with clean labels are selected to train the model. Then, the model is progressively used to select more samples with clean labels. In most cases, consistently high-confidence predictions, during training, are generally associated with uncorrupted labels. This strategy can be considered a correction mechanism as identifying the noisy labels requires training on them in advance. However, it is not clear to what extent the predictions of a noisy classifier (i.e., trained on corrupted labels) are sufficient to recognize samples with corrupted labels.

Compared with existing sampling techniques for supervised learning, our strategy is motivated by two additional insights. The first idea consists of using a protection mechanism against FR, instead of a correction one. In fact, it has been observed that fine-tuning a model by training it on ground-truth labels, once the pretraining phase is performed on random labels, can not reverse the impact of labels’ randomness [15]. Since a correction mechanism can not reverse the effect of labels’ randomness, we opt for a protection mechanism that prioritizes the selection of samples with uncorrupted labels, before using them for training. Our sampling technique is initiated directly after the pretraining phase and exploits two strong criteria to collect a sufficient portion of nodes with reliable clustering assignments. Second, we argue that it is important to control the selection process according to the difference between the first high-confidence and second high-confidence clustering assignment scores. This aspect is quite useful when the labels can be flipped between two similar clusters.

We propose three guidelines to develop our sampling operator \( \Xi \). The first guideline consists of transforming hard clustering assignments into soft assignments. To this end, we compute the matrix \( (p'_{ij})_{ij} \in \mathbb{R}^{N \times K} \). If \( (p_{ij})_{ij} \) is already a soft assignment matrix, then we set \( p'_{ij} = p_{ij} \). If the matrix \( (p_{ij})_{ij} \) is a hard assignment matrix, then we measure the similarity between the embedded points and the clustering representatives according to:

\[
n_{ij} = \frac{\exp(-\frac{1}{2} (z_i - \mu_j) \Sigma_j^{-1} (z_i - \mu_j))}{\sum_{j=1}^{K} \exp(-\frac{1}{2} (z_i - \mu_j) \Sigma_j^{-1} (z_i - \mu_j))},
\]

(14)

where \( \mu_j \) stands for the center of cluster \( C_j^{clus} \), and \( \Sigma_j \) is a diagonal matrix representing the cluster variance. The second guideline consists of extracting the first and second high-confidence assignment scores from matrix \( (p'_{ij})_{ij} \) for each node. The first score associated with \( z_i \) is denoted by \( \lambda_i^1 \):

\[
\lambda_i^1 = \max_{j \in \{1, ..., K\}} (p'_{ij}).
\]

(15)

The second high-confidence assignment score for the embedded representation \( z_i \) is denoted by \( \lambda_i^2 \):

\[
\lambda_i^2 = \max_{j \in \{1, ..., K\}} (p'_{ij} | p'_{ij} < \lambda_i^1).
\]

(16)

The third guideline consists of constructing a set \( \Omega(t) \) that contains nodes, whose clustering assignments at iteration \( t \) are reliable enough to decide to which cluster they belong. Points from \( \Omega \) are selected according to two criteria as described by Eq. (17).

\[
\Omega = \{ i \in \mathcal{V} | \lambda_i^1 \geq \alpha_1 \text{ and } (\lambda_i^1 - \lambda_i^2) \geq \alpha_2 \}.
\]

(17)

First, a node from \( \Omega \) is situated close to its closest cluster representative. Consequently, its first high-confidence assignment score is greater than a threshold \( \alpha_1 \), where \( \alpha_1 \) is a tunable hyper-parameter in the range \([0, 1]\). Second, a point from \( \Omega \) is located far from the borderline between neighboring clusters. Consequently, the difference between the first and second high-confidence assignment scores is greater than a threshold \( \alpha_2 \). We set \( \alpha_1 = \frac{\alpha_2}{2} \). Our sampling operator \( \Xi \) is summarized in Algorithm 1. The computational complexity of Algorithm 1 is \( O(NK^2d) \).

**Algorithm 1. Operator \( \Xi \)**

1: **Input**: Embedded data: \( Z \), Number of clusters: \( K \), First confidence threshold: \( \alpha_1 \), Second confidence threshold: \( \alpha_2 \).
2: **Output**: Embedded representations of decidable nodes: \( Z[\Omega] \).
3: Compute the matrix \( (p'_{ij})_{ij} \in \mathbb{R}^{N \times K} \) according to Eqn. (14).
4: for \( i = 0 \) to \( |\mathcal{X}| \) do
5: \hspace{1em} Compute \( \lambda_i^1 \) according to Eq. (15).
6: \hspace{1em} Compute \( \lambda_i^2 \) according to Eq. (16).
7: end for
8: Construct \( \Omega \) according to Eq. (17).
9: Return \( Z[\Omega] \).

### 4.2 A Correction Mechanism Against FD

Real-world graphs carry edges that connect nodes from different clusters. Reconstructing the input graph structure is not suitable for learning clustering-oriented embeddings. To attenuate FD, we use the embedding of reliable nodes \( \Xi(Z(\theta)) \) to gradually transform the reconstruction objective into a clustering-oriented cost. This can be done by gradually substituting the self-supervision signal \( A_{self} \) with a task-specific signal \( \Upsilon(A, \mathcal{P}(\Xi(Z(\theta))), \Omega) \).

We propose two guidelines for developing the graph transforming operator \( \Upsilon \). The first guideline consists of identifying a centroid node for each cluster. To this end, we compute \( \bar{\mu}_j \), which averages the embedded representations of reliable nodes from cluster \( C_j^{clus} \). Then, for each \( \bar{\mu}_j \), we search for its nearest node, in the embedded space, among the set \( \Omega \). The list of obtained nodes is denoted by \( \mathcal{H} = [i \in \mathcal{V} | i = 1-\text{NN}(\bar{\mu}_j, \Omega) \text{ and } j \in \{1, ..., K\}] \), where 1-NN represents the nearest neighbor algorithm.

The second guideline consists of constructing a new self-supervision signal \( A_{clus} \) based on the original graph structure \( A \). To this end, we start by connecting each node from
\( \Omega \) with its associated centroid from \( \Pi \). Then, we drop edges between nodes from \( \Omega \), which are members of different clusters. As a result, the obtained graph \( A^{self}_{clus} \) contains \( K \) star-shaped subgraphs representing the different clusters. Algorithm 2 summarizes our proposed operator \( \Upsilon \). The worst-case complexity of Algorithm 2 is \( O(N(d+K)+|E|(N+K)) \).

Algorithm 2. Operator \( \Upsilon \)

1: **Input:** Original sparse graph: \( A \), Clustering assignment: \( P \), Set of decidable nodes: \( \Omega \).
2: **Output:** Clustering-oriented self-supervision graph: \( A^{self}_{clus} \).
3: \( \Pi \leftarrow [i \in V | i = 1-NN(\mu_j, \Omega) \text{ and } j \in \{1, ..., K\}] \).
4: \( A^{self}_{clus} \leftarrow A \).
5: for \( i \in \Omega \) do
6: \( k_1 \leftarrow \text{argmax}_k (P[i, k]) \).
7: \( j \leftarrow \Pi[k_1] \).
8: \( k_2 \leftarrow \text{argmax}_k (P[j, k]) \).
9: if \( (j \notin A[i].indices \text{ and } (k_1 = k_2) \lor A[i].indices \text{ indicates the list of nodes connected to node } i) \) then
10: \( A^{self}_{clus}[i, j] \leftarrow 1 \).
11: end if
12: for \( i \in A[i].indices \) do
13: \( k_3 \leftarrow \text{argmax}_l (P[l, k]) \).
14: if \( (l \notin \Omega) \text{ and } (k_1 \neq k_2) \) then
15: \( A^{self}_{clus}[i, l] \leftarrow 0 \).
16: end if
17: end for
18: end for
19: **Return** \( A^{self}_{clus} \).

A protection mechanism against FD can be established by transforming the self-supervision signal \( A \) into a clustering-oriented signal \( \Upsilon(A, P(Z(\theta)), V) \), in a single step. This is done by applying \( \Upsilon \) to the whole set of nodes \( V \), instead of \( \Omega \). We argue that a correction mechanism, which allows FD to take place then gradually attenuates this problem, is a more advantageous solution.

5 EXPERIMENTS

To validate the suitability of our conceptual design and proposed operators, we conduct an extensive experimental protocol.\(^1\) We show that it is possible to substantially improve the clustering performance of several GAE-based clustering models by integrating operators that can control FR and FD. We obtain promising results, which calls for further research in this direction.

5.1 Experimental Settings

Due to the limited number of second-group models, we propose a new approach entitled DGAE from this group. We provide a technical description of this method in Appendix B, available in the online supplemental material. The val-

\[^1\] Our code is available at: https://github.com/nairouz/R-GAE

which, as discussed in Section 2, establishes clustering and embedding learning separately. The clustering process of these models aligns with the abstract formulation in Eq. (1). Specifically, these four models do not optimize the encoding weights using a clustering loss. Instead, they only perform K-means on the latent representations at the end of the training process.

For models in the first category, we perform embedded K-means to obtain the clustering assignments. We feed these assignments to our operator \( \Xi \) to build the set \( \Omega \). Then, we feed this set to the operator \( \Upsilon \) to gradually transform the reconstruction loss into a clustering-oriented objective.

DGAE and GMM-VGAE are members of the second group, which ensures joint clustering and embedding learning. For these models, we feed their clustering assignments to our operator \( \Xi \) to gradually build the set of reliable nodes. We compute the clustering loss only on the set of high-confidence samples. We also exploit the second operator \( \Upsilon \) to gradually transform the reconstruction into a clustering-oriented objective.

For GAE, VGAE, ARGAE, and ARVGAE, we use the publicly available implementations. For GMM-VGAE, we reproduce their reported results by performing our implementation. We keep the original settings (optimizer, hyper-parameters, and architecture) of each model for fairness of comparison. The obtained methods are abbreviated by (R-GAE, R-VGAE, R-ARGAE, R-ARVGAE, R-GMM-VGAE, R-DGAE). “R-D” stands for Rethinking the model \( D \) (i.e., GAE, VGAE, ARGAE, ARVGAE, GMM-VGAE, DGAE) from the perspective of FR and FD. To avoid training instability due to the consistent modification of the self-supervision signal, we update \( \Omega \) and \( A^{self}_{clus} \) every \( M_1 \) and \( M_2 \) iterations, respectively. We train the obtained models until meeting the convergence criterion \(|\Omega| \geq 0.9 \times |V|\). We devote Appendix B, available in the online supplemental material to provide the loss functions and training mechanisms of the different methods. Compared with the original approaches, that is, GAE, VGAE, ARGAE, ARVGAE, GMM-VGAE, and DGAE, three additional hyper-parameters, namely \( M_1, M_2 \) and \( \alpha_1 \), should be specified. The values of these parameters are provided in Appendix D, available in the online supplemental material.

We assess the proposed operators on six benchmark datasets. Our evaluation includes three citation networks (Cora, Citeseer, and Pubmed [40]) and three air-traffic networks (USA, Europe, and Brazil [41]). The nodes of the citation networks correspond to scientific publications and the graph edges capture the citations. The nodes of the air traffic graphs correspond to the airports. The edges of these graphs capture the existence of commercial flights between the airports. The labels indicate the level of activity of each airport. Since the air-traffic networks do not have node attributes, we leverage the one-hot encoding of node degrees to construct the feature matrix \( X \) similar to [42]. For all datasets, \( X \) is row-normalized with the euclidean norm. In Table 1, we summarize the data statistics for the citation graphs. In Table 2, we summarize the data statistics for the air traffic graphs. All experiments are performed under the same hardware and software environments as described in Appendix A, available in the online supplemental material.
In Tables 3, 4, 5, and 6, we report the best $L_u$ values for GMM-VGAE and R-GMM-VGAE on Cora. $\cos$ denotes the cosine similarity between the gradient of $L_u(Z(\theta), P)$ and the gradient of $L_{clus}(Z(\theta), Q')$ is denoted by $\lambda_{FR}(GMM-VGAE)$, whereas $\lambda_{FR}(R-GMM-VGAE)$ denotes the cosine similarity between the gradient of $L_{clus}(Z(\theta), P)$ and the gradient of $L_{clus}(Z(\theta), Q')$. We illustrate both metrics, during training of R-GMM-VGAE and GMM-VGAE, in Figs. 4a and 4b, respectively. To facilitate our analysis, we also provide the normalized cumulative difference between $\lambda_{FR}(R-GMM-VGAE)$ and $\lambda_{FR}(GMM-VGAE)$, during the training of R-GMM-VGAE and GMM-VGAE, in Figs. 4d and 4e, respectively. As a general observation from Figs. 4a, 4b, and 4c, $\lambda_{FR}(GMM-VGAE)$ and $\lambda_{FR}(R-GMM-VGAE)$ start from very high values (close to one). This implies that the unsupervised gradient, at an early training stage, has the same direction as the supervised one. This result is congruent with recent findings, which suggest that training with ground-truth or random labels prioritizes learning simple patterns first at the level of the earlier layers [43, 44]. These simple patterns are not dependent on the target labels [15].

**TABLE 1**

| Dataset     | Cora | Citeseer | Pubmed |
|-------------|------|----------|--------|
| Number of nodes | 2708 | 3327 | 19717 |
| Number of edges | 5429 | 4732 | 44338 |
| Number of features | 1433 | 3703 | 500 |
| Number of classes | 7 | 6 | 3 |

**5.2 Results**

We present the principal results of our experiments in this section. However, due to page limit restriction, we provide further experiments and results in Appendix E (Supplementary Material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3220948).

**Effectiveness:** In Tables 3, 4, 5, and 6, we report the best and average clustering results among three trials on six datasets. For all tables, we mark the best methods in bold and the clustering performances in %. For fairness of comparison, we ensure that each couple of methods $D$ and $R-D$ share the same pretraining weights before starting the clustering phase. Table 3 provides the best clustering performances on three citation networks. From this table, we observe that the second GAE group methods yield considerably better results than methods from the first group. These results confirm that performing joint clustering and embedding learning is advantageous to the clustering task. Among the first group, we can see that (R-GAE, R-VGAE, R-ARGAE, R-ARVGAE) generally have better ACC, NMI, and ARI compared with their counterparts (GAE, VGAE, ARGA, ARGVAE). The embedded representations of (GAE, VGAE, ARGA, ARVGAE) are optimized using the reconstruction objective. These methods do not suffer from FR and FD. By gradually transforming the graph reconstruction into a clustering-oriented loss, during the training process, (R-GAE, R-VGAE, R-ARGAE, R-ARVGAE) make the embedded representations more clustering-oriented. Among the second group, we observe that (R-GMM-VGAE, R-DGAE) outperform their counterparts (GMM-VGAE, DGAE) consistently by a significant margin. To confirm these results, we extend the performed experiments to three additional datasets as shown in Table 5. Our results offer strong evidence that the proposed operators can improve the clustering effectiveness of GAE models in terms of ACC, NMI, and ARI. Since this manuscript aims at investigating the impact of FR and FD, we focus on (R-GMM-VGAE, R-DGAE) and their counterparts (GMM-VGAE, DGAE) in the subsequent experiments. Moreover, we provide a comprehensive comparison with several recent graph clustering methods in Appendix E (Supplementary Material, available online).

**Efficiency:** In Table 7, we compare (R-GMM-VGAE, R-DGAE) with their counterparts (GMM-VGAE, DGAE) in terms of run-time. We report the best, the mean, and the variance in execution time over ten trials. Although Pubmed has almost ten times more edges and features than Cora and Citeseer, we observe that the difference in execution time between (R-GMM-VGAE, R-DGAE) and their counterparts (GMM-VGAE, DGAE) remains considerably small on Pubmed. In accordance with the provided complexity analysis for Algorithms 1 and 2, our results confirm that the designed operators do not cause any significant overhead in execution time, compared with the original models.

**Feature Randomness:** In this part, we discuss the evolution of $\lambda_{FR}$ values for GMM-VGAE and R-GMM-VGAE on Cora. The cosine similarity between the gradient of $L_{clus}(Z(\theta), P)$ and the gradient of $L_{clus}(Z(\theta), Q')$ is denoted by $\lambda_{FR}(GMM-VGAE)$, whereas $\lambda_{FR}(R-GMM-VGAE)$ denotes the cosine similarity between the gradient of $L_{clus}(Z(\theta), P)$ and the gradient of $L_{clus}(Z(\theta), Q')$. We illustrate both metrics, during training of R-GMM-VGAE and GMM-VGAE, in Figs. 4a and 4b, respectively. To facilitate our analysis, we also provide the normalized cumulative difference between $\lambda_{FR}(R-GMM-VGAE)$ and $\lambda_{FR}(GMM-VGAE)$, during the training of R-GMM-VGAE and GMM-VGAE, in Figs. 4d and 4e, respectively. As a general observation from Figs. 4a, 4b, and 4c, $\lambda_{FR}(GMM-VGAE)$ and $\lambda_{FR}(R-GMM-VGAE)$ start from very high values (close to one). This implies that the unsupervised gradient, at an early training stage, has the same direction as the supervised one. This result is congruent with recent findings, which suggest that training with ground-truth or random labels prioritizes learning simple patterns first at the level of the earlier layers [43, 44]. These simple patterns are not dependent on the target labels [15].

For the first experiment (Figs. 4a and 4d), we train R-GMM-VGAE and we report $\lambda_{FR}(GMM-VGAE)$, $\lambda_{FR}(R-GMM-VGAE)$, and the normalized cumulative difference between both of them. We can see that there are two stages. The first stage ranges from iteration 0 to 60, and the second stage ranges from iteration 60 to 140. For the first stage, we observe that $\lambda_{FR}(R-GMM-VGAE)$ is higher than $\lambda_{FR}(GMM-VGAE)$.
This result is confirmed by observing the cumulative difference between $\Lambda_{FD}(\text{R-GMM-VGAE})$ and $\Lambda_{FD}(\text{GMM-VGAE})$ in Fig. 4d, which has a pronounced increasing tendency. These results demonstrate the ability of our operator $\Xi$ to reduce FR during the first stage. For the second stage (from iteration 60 to 140 of Fig. 4a), the blue and green curves become closer to each other. This observation is confirmed by a lower slope for the curve of Fig. 4d compared with the slope of the same curve for the first stage (i.e., between iterations 0 and 60 of Fig. 4d). At this point, $\Omega$ gradually approaches $\nu$. Therefore, $\Lambda_{FD}(\text{R-GMM-VGAE})$ becomes approximately equal to $\Lambda_{FD}(\text{GMM-VGAE})$.

For the second experiment (Figs. 4b and 4e), we train GMM-VGAE and we report $\Lambda_{FD}(\text{GMM-VGAE})$, $\Lambda_{FD}(\text{R-GMM-VGAE})$, and the normalized cumulative difference between both of them. We observe that $\Lambda_{FD}(\text{R-GMM-VGAE})$ is consistently close to 1. From Fig. 4e, we can see that the cumulative difference between $\Lambda_{FD}(\text{R-GMM-VGAE})$ and $\Lambda_{FD}(\text{GMM-VGAE})$ has almost a constant slope. These results suggest that $\Xi$ can consistently select a sufficient amount of reliable nodes even after learning based on unreliable nodes. Thus, $\Xi$ is capable of playing the role of a protection mechanism against FR.

For the third experiment (Figs. 4c and 4f), we train GMM-VGAE and report $\Lambda_{FD}(\text{GMM-VGAE})$, we train R-GMM-VGAE and report $\Lambda_{FD}(\text{R-GMM-VGAE})$, and we finally report the normalized cumulative difference between both of them. We can see that there are three stages. The first stage ranges from iteration 0 to 50, the second stage ranges from iteration 50 to 100, and the third stage ranges from iteration 100 to 140. For the first stage, we observe that R-GMM-VGAE outperforms GMM-VGAE in terms of $\Lambda_{FD}$ thanks to our operator $\Xi$. For the second stage, we observe that GMM-VGAE yields better results than R-GMM-VGAE in terms of $\Lambda_{FD}$. To reduce FR, R-GMM-VGAE transforms the reconstruction loss into a clustering-oriented loss. However, eliminating the reconstruction gives rise to FR. Unlike R-GMM-VGAE, GMM-VGAE maintains the reconstruction loss during the second stage, which is considered an implicit mechanism against FR. Fig. 4f shows clearly the trade-off between FR and FD. Although both models have reduced the same amount of FR, delaying the effect of FR has a favorable impact on the clustering performance. For the third stage, both models tie together. This experiment shows that using a protection mechanism delays the effect of FR and does not prevent it from taking place. By delaying the effect of randomness using a protection mechanism, it is possible to improve the clustering performance considerably.

### Feature Drift

In this part, we discuss the evolution of $\Lambda_{FD}$ values for GMM-VGAE and R-GMM-VGAE on Cora. The cosine similarity between the gradient of $L_{bce}(\hat{A}(Z(\theta)), A)$ and the gradient of $L_{bce}(A(Z(\theta)), Y)$ is denoted by $\Lambda_{FD}(\text{GMM-VGAE}_{bce})$, whereas $\Lambda_{FD}(\text{R-GMM-VGAE}_{bce})$ denotes the cosine similarity between the gradient of $L_{bce}(\hat{A}(Z(\theta)), \Omega)$ and the gradient of $L_{bce}(A(Z(\theta)), Y(A, Q'(Z(\theta)), V))$. We illustrate both metrics, during training of R-GMM-VGAE and GMM-VGAE, in Figs. 5a and 5b, respectively. To facilitate our analysis, we also provide the normalized cumulative difference between $\Lambda_{FD}(\text{R-GMM-VGAE})$ and $\Lambda_{FD}(\text{GMM-VGAE}_{bce})$, during training of R-GMM-VGAE and GMM-VGAE, in Figs. 4d and 4e, respectively. As a general observation from Figs. 5a, 5b, and 5c, $\Lambda_{FD}(\text{R-GMM-VGAE})$ and $\Lambda_{FD}(\text{GMM-VGAE})$ start from very high values (close to one) then gradually decrease. This implies that the
unsupervised gradient, at an early training stage, has the same direction as the supervised one. A recent body of work [43], [44] has shown that a neural network learns simple patterns first using the early layers. In another work, the authors of [45] have shown that these simple patterns can be learned through self-supervision just as well as through real supervision (with ground-truth labels). Thus, optimizing a supervised objective function has the same effect (learning low-level patterns) as optimizing a self-supervised objective function for the first few iterations.

For the first experiment (Figs. 5a and 5d), we train R-GMM-VGAE and we report $L_{FD}(R\text{-}GMM-VGAE)$, $L_{FD}(GMM-VGAE)$, and the normalized cumulative difference between both of them. We can see that there are two stages. The first stage ranges from iteration 0 to 40, and the second stage ranges from iteration 40 to 140. For the first stage, we observe that $L_{FD}(R\text{-}GMM-VGAE)$ values are very close to $L_{FD}(GMM-VGAE)$ values. A possible explanation is that $\gamma$ can only affect a small part of the self-supervision graph at the beginning, and most of the graph remains identical to $A$. Furthermore, we observe that $L_{FD}(R\text{-}GMM-VGAE)$ is decreasing rapidly for this stage. This aspect is desirable. In fact, $\gamma$ allows FD to occur at the beginning to counter random projections. From Fig. 5d, we can see that

### TABLE 6
Mean and Standard Deviation of the Evaluation Metrics for the Original and Proposed GAE Models on Air-Traffic Datasets

| Method      | USA Air-Traffic | Europe Air-Traffic | Brazil Air-Traffic |
|-------------|-----------------|--------------------|--------------------|
|             | ACC  | NMI  | ARI   | ACC  | NMI  | ARI   | ACC  | NMI  | ARI   |
| GMM-VGAE    | 47.2 ± 0.9 | 21 ± 0.8 | 12.7 ± 0.5 | 52.3 ± 1.0 | 29.2 ± 1.8 | 22.6 ± 1.5 | 69.0 ± 1.6 | 43.7 ± 2.6 | 38.8 ± 3.2 |
| R-GMM-VGAE  | 50.4 ± 0.59 | 22.6 ± 0.5 | 15.2 ± 0.6 | 56.4 ± 1.3 | 31.2 ± 0.78 | 25.3 ± 0.8 | 71.8 ± 1.6 | 45.0 ± 2.7 | 41.6 ± 3.4 |
| DGAE        | 45.8 ± 0.6 | 28.1 ± 0.2 | 18.2 ± 0.3 | 53.2 ± 0.5 | 33.1 ± 0.2 | 23.1 ± 0.2 | 70.7 ± 0.4 | 48.1 ± 1.0 | 39.9 ± 1.3 |
| R-DGAE      | 51.3 ± 0.4 | 24.4 ± 0.4 | 16.2 ± 0.4 | 56.7 ± 0.7 | 32.2 ± 1.1 | 24.3 ± 0.8 | 74.1 ± 0.3 | 52.4 ± 1.3 | 45.7 ± 0.6 |

### TABLE 7
Execution Time (in Seconds) of the Couples (GMM-VGAE, R-GMM-VGAE) and (DGAE, R-DGAE)

| Method      | Cora     | Citeseer  | Pubmed   |
|-------------|----------|-----------|----------|
|             | Best     | Mean      | Variance | Best     | Mean      | Variance | Best     | Mean      | Variance |
| GMM-VGAE    | 17.135   | 17.703    | 0.530    | 36.269   | 36.442    | 1.436    | 1341.190 | 1348.960 | 16.056   |
| R-GMM-VGAE  | 21.928   | 24.509    | 2.589    | 40.084   | 41.910    | 2.884    | 1457.188 | 1477.405 | 155.492  |
| DGAE        | 19.298   | 20.179    | 0.644    | 38.074   | 38.226    | 0.012    | 1067.301 | 1076.431 | 33.446   |
| R-DGAE      | 28.981   | 31.053    | 1.464    | 51.363   | 52.976    | 1.850    | 1192.913 | 1215.241 | 361.036  |

Fig. 4. Performance of R-GMM-VGAE and GMM-VGAE in terms of $L_{FR}$ on Cora. Blue line: $\lambda_{FR}$ values of R-GMM-VGAE, during training of R-GMM-VGAE. Green line: $\lambda_{FR}$ values of GMM-VGAE, during training of R-GMM-VGAE. Gold line: $\lambda_{FR}$ values of R-GMM-VGAE, during training of GMM-VGAE. Red line: $\lambda_{FR}$ values of GMM-VGAE, during training of GMM-VGAE. Purple line: normalized cumulative difference between $\lambda_{FR}$ values of R-GMM-VGAE and $\lambda_{FR}$ values of GMM-VGAE.
the cumulative difference between $\Lambda_{FD}(R$-GMM-VGAE) and $\Lambda_{FD}(GMM-VGAE)$ has a low slope for the first stage. This result confirms that our operator $\Upsilon$ allows FD to take place, during the first stage. For the second stage, we observe that $\Lambda_{FD}(R$-GMM-VGAE) is increasing slowly between iterations 40 and 60. After allowing FD to occur, during the first stage, $\Upsilon$ gradually attenuates this problem during the second stage. From Fig. 5d, we can see that the cumulative difference between $\Lambda_{FD}(R$-GMM-VGAE) and $\Lambda_{FD}(GMM-VGAE)$ has a pronounced increasing tendency compared with the first phase. After allowing FD to occur, $\Upsilon$ gradually attenuates this problem during the second stage.

For the second experiment (Figs. 5b and 5e), we train GMM-VGAE and we report $\Lambda_{FD}(GMM-VGAE)$, $\Lambda_{FD}(R$-GMM-VGAE), and the normalized cumulative difference between both of them. From Fig. 5e, we can see that the cumulative difference between $\Lambda_{FD}(R$-GMM-VGAE) and $\Lambda_{FD}(GMM-VGAE)$ has a pronounced increasing tendency starting from iteration 40. This result suggests that $\Upsilon$ can consistently construct a reliable self-supervision signal even after learning based on unreliable nodes. Additionally, we observe a decreasing tendency of $\Lambda_{FR}$ between iterations 0 and 100. After 100 iterations, the two curves of $\Lambda_{FD}$ in Fig. 5b oscillate around a horizontal line (indicating the stability of FD). The absence of a considerable time slot, where $\Lambda_{FD}(GMM-VGAE)$ achieves a clear increasing tendency, suggests that GMM-VGAE does not have any implicit or explicit mechanism to reduce FD. Based on the same experiment, we can see that $\Lambda_{FD}(GMM-VGAE)$ can reach very low values compared with $\Lambda_{FR}(GMM-VGAE)$ (see Fig. 4b). In addition to that, we observe that $\Lambda_{FD}(GMM-VGAE)$ has more pronounced fluctuations than $\Lambda_{FR}(GMM-VGAE)$. While GMM-VGAE does not have any explicit mechanism

| Method      | Protection ACC | Protection NMI | Correction ACC | Correction NMI |
|-------------|----------------|----------------|----------------|----------------|
| R-GMM-VGAE  | 76.7           | 57.3           | 74.5           | 53.9           |
| R-DGAE      | 73.7           | 56.0           | 71.1           | 52.0           |

| Method      | ACC  | NMI  | ARI  |
|-------------|------|------|------|
| R-GMM-VGAE  | 73.4 | 52.1 | 51.6 |
| R-DGAE      | 71.3 | 54.5 | 50.4 |
against FR or FD, the reconstruction loss is an implicit mechanism against FR.

For the third experiment (Figs. 5c and 5f), we train GMM-VGAE and report $\Lambda_{FD}(\text{GMM-VGAE})$, we train R-GMM-VGAE and report $\Lambda_{FD}(\text{R-GMM-VGAE})$, and we finally report the normalized cumulative difference between both of them. We observe that R-GMM-VGAE considerably outperforms GMM-VGAE in terms of $\Lambda_{FD}$. More interestingly, while R-GMM-VGAE can attenuate FD after the initial decrease of $\Lambda_{FD}$, GMM-VGAE falls short of this capacity.

Protection Versus Correction: In Table 8, we compare between a protection mechanism and a correction mechanism against FR, during the training of R-GMM-VGAE and R-DGAE on Cora. A protection mechanism is established by initiating the sampling technique directly after the pre-training phase. For the correction case, we delay the sampling technique for different epochs (10, 30, 50, 100, and 150) to allow FR to occur. This experiment aims to test if a correction mechanism can reverse the effect of labels’ randomness. As we can see from Table 8, the protection strategy yields better results than the correction approaches for both models. Moreover, further delay of correction is generally associated with lower clustering performance. These results show that a correction mechanism cannot reverse the effect of labels’ randomness. In Table 9, we compare between a protection mechanism and a correction mechanism against FD, during the training of R-GMM-VGAE and R-DGAE on Cora. A protection mechanism is established by transforming the self-supervision signal $A$ into a clustering-oriented signal $\mathcal{Y}(A, P(\mathcal{Z}(\theta)), \nu)$, in a single step. This is done by applying $\mathcal{Y}$ to the whole set of nodes $\nu$, instead of $\Omega$, to eliminate the reconstruction. We observe that the correction strategy yields better results than the protection approach for both models. We conclude that a correction mechanism, which allows FD to take place then gradually attenuates this problem, is a more advantageous solution.

One Confidence Threshold versus Two Confidence Thresholds: In this part, we perform an ablation study to investigate the performance contribution of $\mathcal{Y}$. Our investigation includes four cases: ablation of the sampling criteria related to $\alpha_1$, ablation of the sampling criteria related to $\alpha_2$, ablation of both (i.e., eliminating the operator $\Xi$), and no ablation. As shown in Table 10, the obtained results show the importance of using two criteria for selecting reliable nodes. Specifically, we observe that ablating the requirement related to $\alpha_2$ leads to a degradation in performance. In fact, $\alpha_2$ helps in excluding points, which are situated near the borderline of two similar clusters.

Adding Edges versus Dropping Edges: In this part, we perform an ablation study to investigate the performance contribution of $\mathcal{Y}$. Our investigation includes four cases: ablation of “drop_edge”, ablation of “add_edge”, ablation of both (i.e., eliminating the operator $\mathcal{Y}$), no ablation. As shown in Table 11, the obtained results show the importance of “add_edge” and “drop_edge” operations for building a reliable self-supervision signal $A_{clus}^{rel}$.

6 CONCLUSION

In this manuscript, we advocate a new vision for building GAE-based clustering models from the perspective of Feature Randomness and Feature Drift. We start by introducing a new conceptual design that gradually reduces Feature Drift without causing an abrupt rise in random features. Our strategy depends on two operators. In this regard, we design a sampling function $\mathcal{Y}$ that triggers a protection mechanism against random projections. Moreover, we propose a function $\mathcal{Y}$ that triggers a correction mechanism against Feature Drift. As a key advantage, $\mathcal{Y}$ and $\mathcal{Y}$ can be easily tailored to existing GAE-based clustering models. Experiments on standard benchmarks demonstrate that our operators improve the clustering performance. Furthermore, our results show that: (1) $\mathcal{Y}$ effectively delays the impact of Feature Randomness, and (2) $\mathcal{Y}$ allows Feature Drift to occur then gradually reduces this problem. Our operators can be viewed as the first initiative to control Feature Randomness and Feature Drift for GAE-based clustering models. For future work, we plan to investigate the extensibility of our operators to multiplex graphs, in which each couple of nodes can be connected by multiple edges.
REFERENCES

[1] H. Cai, V. W. Zheng, and K. C.-C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[2] P. Cui, X. Wang, J. Pei, and W. Zhu, “A survey on network embedding,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 5, pp. 833–852, Sep. 2018.

[3] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, “Learning deep representations for graph clustering,” in Proc. Conf. Assoc. Adv. Intell. Intell., 2014, pp. 1293–1299.

[4] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in Proc. World Wide Web Conf., 2015, pp. 1067–1077.

[5] D. Cai, X. He, J. Han, and T. S. Huang, “Graph regularized non-negative matrix factorization for data representation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

[6] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Discov. Data Mining, 2016, pp. 855–864.

[7] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.

[8] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017.

[9] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” in Proc. Int. Conf. Neural Inf. Process. Syst. Workshop, 2016, pp. 1–3.

[10] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, “Adversarially regularized graph autoencoder for graph embedding,” in Proc. Int. Joint Conf. Artif. Intell., 2018, pp. 2609–2615.

[11] B. Hui, P. Zhu, and Q. Hu, “Collaborative graph convolutional networks: Unsupervised learning meets semi-supervised learning,” in Proc. Conf. Assoc. Adv. Intell. Intell., 2020, pp. 4215–4222.

[12] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, “Attributed graph clustering: A deep attentional embedding approach,” in Proc. Int. Joint Conf. Artif. Intell., 2019, pp. 2609–2615.

[13] X. Guo, L. Gao, X. Liu, and J. Yin, “Improved deep embedded clustering with local structure preservation,” in Proc. Int. Joint Conf. Artif. Intell., 2017, pp. 1753–1759.

[14] N. M. Minh, M. Sugiyama, and R. Ksantini, “Adversarial deep clustering: On a better trade-off between feature randomness and feature drift,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 04, pp. 1603–1617, Apr. 2022.

[15] H. Maenenn et al., “What do neural networks learn when trained with random labels?” in Proc. Int. Conf. Neural Inf. Process. Syst., 2020, pp. 1028–1038.

[16] J. Franklin, D. J. Schwab, and A. S. Morcos, “The early phase of neural network training,” in Proc. Int. Conf. Learn. Representations, 2020.

[17] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, “MGAE: Marginalized graph autoencoder for graph clustering,” in Proc. Conf. Int. Knowl. Manage., 2017, pp. 889–898.

[18] A. Anasuni, A. Laio, J. H. Macke, and D. Zoccolan, “Intrinsic dimension of data representations in deep neural networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 6111–6122.

[19] H. W. Kuhn, “The hungarian method for the assignment problem,” Nav. Res. Logistics Quart., vol. 2, no. 1/2, pp. 83–97, 1955.

[20] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rereading generalization,” in Proc. Int. Conf. Learn. Representations, 2017.

[21] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minimizers,” in Proc. Int. Conf. Learn. Representations, 2017.

[22] X. Ma et al., “Dimensionality-driven learning with noisy labels,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 3355–3364.

[23] G. Li, F. Cui, Y. Zhou, C. Yang, and Z. Liu, “Adaptive graph encoder for attributed graph embedding,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 976–985.

[24] Y. You, T. Chen, Z. Wang, and Y. Shen, “When does self-supervision help graph convolutional networks?,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 10 871–10 880.
Mohamed Bouguessa received the MSc and PhD degrees from the University of Sherbrooke, Quebec, Canada, in 2005 and 2009, respectively. He is currently an associate professor of computer science with the University of Quebec at Montreal (UQAM), Montreal, Quebec, Canada. His research covers a variety of data mining related topics. His current research projects include clustering, dynamic networks and graph neural networks.

Mohamed Fawzi Touati is currently working toward the MSc degree with the University of Quebec at Montreal (UQAM), Montreal, Quebec, Canada. His research projects include deep clustering and graph neural networks.

Riadh Ksantini received the bachelor’s degree in applied mathematics and computer science from the Faculte des Sciences, Monastir, Tunisia, in 2000 and the MSc and PhD degrees in computer science from the University of Sherbrooke, Sherbrooke, QC, Canada, in 2003 and 2007, respectively. He is currently an associate professor with the Department of Computer Science, University of Bahrain, Kingdom of Bahrain. His research interests include computer vision, machine/deep learning, and pattern recognition.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csl.