Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051816

ARTICLE

Efficient Clustering Network Based on Matrix Factorization

Jieren Cheng[1,3], Jimei Li[1,3,*], Faqiang Zeng[1,3], Zhicong Tao[1,3] and Yue Yang[2,3]

[1]School of Computer Science and Technology, Hainan University, Haikou, 570228, China

[2]School of Cyberspace Security, Hainan University, Haikou, 570228, China

[3]Hainan Blockchain Technology Engineering Research Center, Haikou, 570228, China

*Corresponding Author: Jimei Li. Email: 22220854000328@hainanu.edu.cn

Received: 15 March 2024 Accepted: 21 May 2024 Published: 18 July 2024

ABSTRACT

Contrastive learning is a significant research direction in the field of deep learning. However, existing data augmentation methods often lead to issues such as semantic drift in generated views while the complexity of model pre-training limits further improvement in the performance of existing methods. To address these challenges, we propose the Efficient Clustering Network based on Matrix Factorization (ECN-MF). Specifically, we design a batched low-rank Singular Value Decomposition (SVD) algorithm for data augmentation to eliminate redundant information and uncover major patterns of variation and key information in the data. Additionally, we design a Mutual Information-Enhanced Clustering Module (MI-ECM) to accelerate the training process by leveraging a simple architecture to bring samples from the same cluster closer while pushing samples from other clusters apart. Extensive experiments on six datasets demonstrate that ECN-MF exhibits more effective performance compared to state-of-the-art algorithms.

KEYWORDS

Contrastive learning; clustering; matrix factorization

1 Introduction

Due to the potent representation learning capabilities of graph data, Graph Neural Networks (GNNs) have successfully permeated various domains, encompassing node classification [1], graph classification [2], time series analysis, knowledge graphs, and clustering [3]. Within the diverse landscape of graph learning, deep graph clustering [4] emerges as a fundamental yet challenging unsupervised task, marking a recent focal point of research interest. The exploration of deep graph clustering methods encompasses various learning mechanisms. Generative methods leverage generative models to characterize the distribution of graph data, achieving effective clustering of graphs [5–9]. Adversarial methods [10,11] introduce the concept of adversarial training, enhancing clustering performance through the interplay between a generator and a discriminator. Contrastive methods [12–16], on the other hand, propel the development of deep graph clustering by learning the similarity and dissimilarity between samples. Our method falls into the category of multi-view [17] contrastive learning, aligning with the latter approach.

Existing methods generate augmented views of the same nodes through graph augmentation. However, existing studies [18,19] indicate: 1) Due to the inherent characteristics of contrastive learning, sensitivity to noise and incorrect labels can lead to semantic drift and indistinguishable positive samples when inappropriate data augmentation techniques such as edge removal, noise addition, diffusion [13], or masking are used. 2) Due to the computational cost of matrix factorization itself, existing methods still face limitations in handling large sparse datasets. 3) Model pre-training typically requires extensive data for optimal performance. Fine-tuning pre-trained models necessitates large-scale labeled data, which may not be suitable for tasks with limited annotated data. Moreover, due to differences between tasks, transferring pre-trained models to specific tasks can be more complex and challenging.

To address the aforementioned issues, we propose an Efficient Clustering Network based on Matrix Factorization (ECN-MF). The main idea behind this approach is to design a batched low-rank Singular Value Decomposition algorithm and a Mutual Information-Enhanced Clustering Guidance Module. This aims to extract crucial information from the data while better preserving the original information in the embedded data, reducing information loss, and improving the model's generative capabilities. Specifically, in terms of data augmentation, we introduce a batched low-rank Singular Value Decomposition algorithm that decomposes the attribute matrix of large datasets into smaller modules. This allows better exploration of major variation patterns and important information in sparse and large datasets. In the network architecture, we employ a pseudo-siamese neural network with the same structure but without parameter sharing. This enables the model to better capture unique information from each view, thereby enhancing clustering performance. Additionally, we design a Mutual Information-Enhanced Clustering Guidance Module to ensure better preservation of original data information, reducing information loss, and enhancing the model's generative capabilities. It brings samples from the same cluster closer while pushing samples from other clusters apart, further improving clustering performance. The main contributions of this work are summarized as follows:

1. We propose an Efficient Clustering Network based on Matrix Factorization (ECN-MF), which does not require pre-training, thus alleviating the challenges of model pre-training and complex model transfer.
2. We propose a batched low-rank Singular Value Decomposition (SVD) algorithm to address the resource-intensive nature of the SVD algorithm in large sparse datasets. This method not only avoids losing important information during data augmentation but also effectively extracts latent information from the data.
3. We designed a Mutual Information-Enhanced Clustering Module (MI-ECM). It enhances the discriminative capability of the network while ensuring better retention of the original data information in the embedded data.
4. Experimental results demonstrate that our proposed method outperforms existing methods in handling challenges such as large sparse datasets and model transfer complexity.

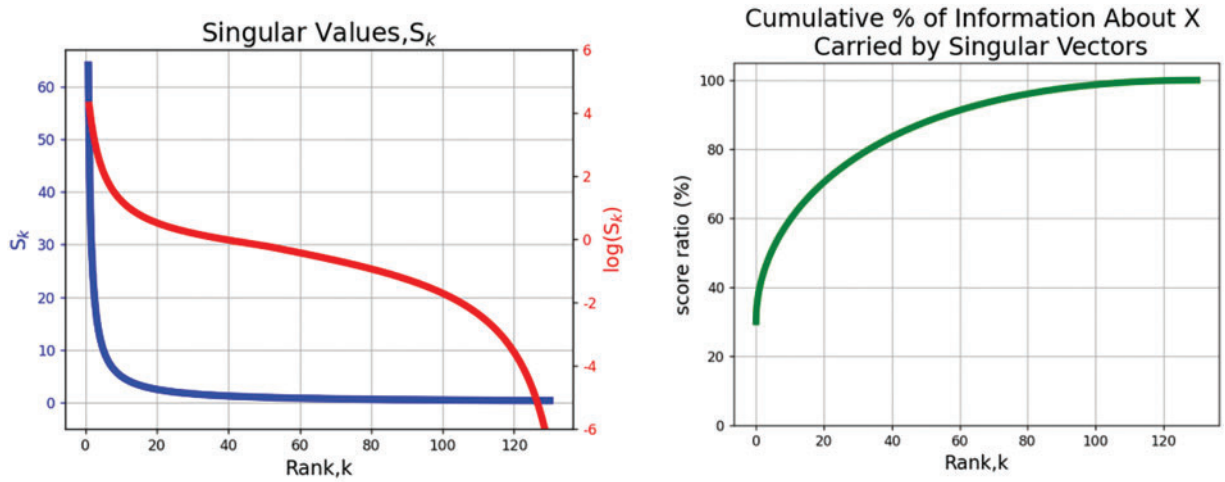
The remaining sections of this paper are organized as follows. Section 2 reviews relevant literature on matrix factorization and contrastive deep graph clustering. In Section 3, we provide detailed explanations of the symbols used, the batched low-rank Singular Value Decomposition algorithm, our network structure, and the Mutual Information-Enhanced Clustering Guidance Module. Section 4 presents the results of our method tested on five datasets. Finally, Section 5 concludes the paper.

2 Related Work

2.1 Matrix Factorization

Matrix factorization (decomposition) is the process of breaking down a matrix into the product of several matrices. This can involve techniques such as triangular factorization, full rank factorization, orthogonal triangle decomposition, Jordan decomposition, and Singular Value Decomposition. In our case, we primarily utilize Singular Value Decomposition to process data. Singular Value Decomposition allows for the representation of a relatively complex matrix as the product of smaller and simpler matrices. These smaller matrices describe the essential characteristics of the original matrix. Singular Value Decomposition is applicable to any matrix, making it adaptable to the features of current attribute information. It finds applications in various fields such as signal processing, statistics, natural language processing, and more. In recommendation systems, Singular Value Decomposition is widely applied in collaborative filtering and matrix completion algorithms. Through Singular Value Decomposition, a user-item rating matrix can be reduced to a low-dimensional latent factor matrix, extracting latent features of users and items for recommendation purposes.

Singular Value Decomposition has significant potential value in processing raw data. The singular values decrease exponentially with rank, with early singular values much larger than later ones, as shown in Fig. 1. By applying low-rank Singular Value Decomposition to the data, we can capture the essential features of the data, enhance data representation, and improve the performance of multi-view clustering. Based on this, we have optimized and improved the Singular Value Decomposition method to be suitable for large datasets, providing higher computational efficiency and scalability.



(a) **(b)**

Figure 1: (a): Singular values exhibit exponential decay with rank, where the initial singular values are significantly larger than the subsequent ones. (b): All information of **X** is encoded in all singular values until ***k***. The majority of information is encoded in the first singular vector returned by Singular Value Decomposition

2.2 Contrastive Deep Graph Clustering

In recent years, contrastive learning has achieved remarkable success in the fields of images [20–23] and graphics [24–26], inspiring extensive research on contrastive deep graph clustering methods

[12–16]. The clustering performance of these methods is primarily influenced by three key factors: data augmentation, network architecture, and the handling of positive and negative sample pairs. Taking these factors into account, we summarize the distinctions between our proposed ECN-MF and other contrastive deep graph clustering methods.

2.2.1 Data Augmentation

Data augmentation techniques are pivotal in the realm of deep graph-contrastive clustering. Current methods, such as edge removal, diffusion, masking, and noise addition, introduce varying degrees of perturbation to the original data. While these approaches help mitigate over-smoothing during iterative training of graph neural networks, they carry the risk of losing crucial information. Inappropriate data augmentation may lead to semantic drift and indistinguishable positive samples, resulting in suboptimal clustering performance. For instance, reference [13] utilizes diffusion matrices as augmented graphs, while Self-supervised contrastive attributed graph clustering (SCAGC) perturbs graph structure by randomly adding or removing edges. Reference [15] and SCAGC enhance node attributes through attribute perturbation. However, reference [27] has highlighted the risk of semantic drift with improper data augmentation. To address this challenge, we propose a novel enhancement approach. Unlike existing methods, ECN-MF leverages batch low-rank Singular Value Decomposition methods to extract crucial attribute information, filter out noisy data, and construct two augmented views of the same node without compromising the original structure.

2.2.2 Network Architecture

In terms of network architecture, SCAGC, reference [28] uses a shared Graph Convolutional Networks (GCNs) encoder to encode nodes. However, conventional GCNs encoders entangle transformation and aggregation operations during the training process, resulting in high time costs. To address this issue, we employ two separate Multilayer Perceptrons (MLPs) to encode the node attributes of the two views. These two MLPs have the same architecture but do not share parameters, ensuring that the node embeddings for the two views contain different semantic information.

2.2.3 Handling of Positive and Negative Sample Pairs

In contrastive methods, the handling of positive and negative sample pairs is crucial. Contrastive methods bring positive samples together while pushing negative samples apart. Therefore, the quality of positive and negative sample pairs significantly influences the performance of contrastive methods. Specifically, reference [13] generates negative samples by randomly shuffling features and designs the InfoMax loss to maximize cross-view mutual information. Reference [12] distinguishes between similar and dissimilar nodes using cross-entropy loss. Subsequently, SCAGC randomly selects samples from different clusters to improve the quality of negative samples. They also design a contrastive clustering loss to maximize the consistency between representations from the same cluster. Both references [14] and [16] utilize the infoNCE loss to attract positive sample pairs and separate negative sample pairs. While their approaches have shown effectiveness, they still depend on a well-pretrained model to choose high-quality positive and negative samples. To address this issue, we propose a MI-ECM to bring samples from the same cluster closer while pushing samples from different clusters apart, thereby enhancing the discriminative ability of sample pairs.

3 Method

In this section, we introduce a novel Efficient Clustering Algorithm based on Matrix Factorization (ECN-MF). The aim is to enhance clustering performance by leveraging matrix factorization in a way that captures unique information from each view with-out disrupting the original structure. The overall framework of ECN-MF is illustrated in Fig. 2. In the following sections, we will provide a detailed explanation of the pro-posed ECN-MF.

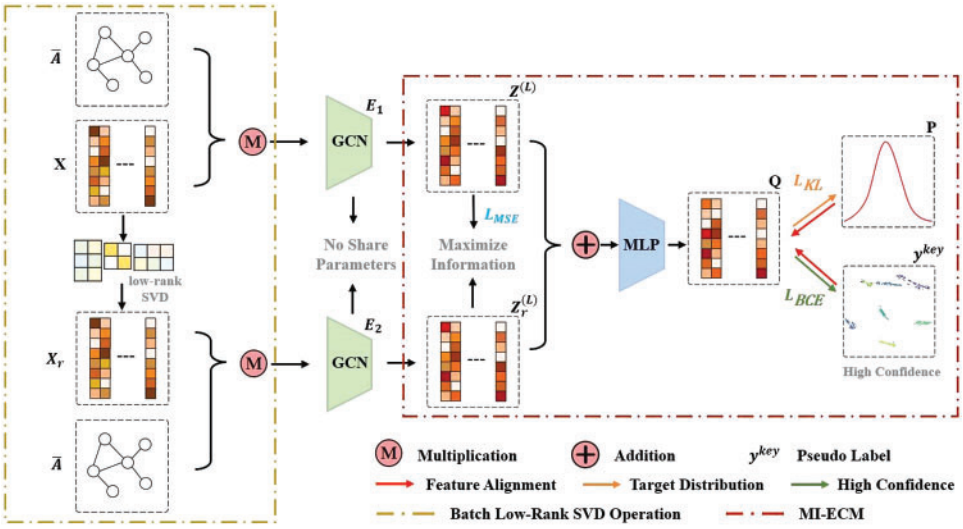


Figure 2: Schematic diagram of the Efficient Clustering Network based on Matrix Factorization. The network consists of three main parts: the batched low-rank Singular Value Decomposition (SVD) on the left, the pseudo-siamese neural network in the middle, and the Mutual Information-Enhanced Clustering Guidance Module on the right. Here, $\overline{\mathbf{A}}$ is the normalized adjacency matrix, $\mathbf{X}$ is the attribute matrix, and $\mathbf{X}_r$ is its corresponding low-rank attribute matrix. $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$ represent the first and second encoders, $\mathbf{Z}^{(L)}$ and $\mathbf{Z}_r^{(L)}$ represent the embedding information of the first and second encoders. $\mathbf{Q}$ is the probability matrix, and $\mathbf{P}$ is the target distribution

3.1 Notations and Problem Definition

In an undirected graph $G = \{\mathbf{X}, \mathbf{A}\}$, let $V = \{v_1, v_2, \ldots, v_N\}$ be a set containing N nodes with K classes, and E be the set of edges. $\mathbf{X} \in \mathrm{R}^{N\times D}$ represents the attribute matrix, and $\mathbf{A} \in \mathrm{R}^{N\times N}$ represents the original adjacency matrix. The degree matrix is denoted as $\mathbf{D} = diag\,(d_1, d_2, \ldots, d_N) \in \mathrm{R}^{N\times N}$, where $d_i = \Sigma_{(v_i,v_j)\in\varepsilon} a_{ij}$. Normalizing the original adjacency matrix $\mathbf{A}$ to $\overline{\mathbf{A}} \in \mathrm{R}^{N\times N}$ is achieved by computing $\mathbf{D}^{(-1)}\,(\mathbf{A} + \mathbf{I})$, where $\mathbf{I} \in \mathrm{R}^{N\times N}$ is the identity matrix. Table 1 summarizes these symbols.

Table 1: Description of the used notations

Notation	Meaning
$\mathbf{X} \in R^{N\times D}$	Attribute matrix
$\mathbf{X}_r \in R^{N\times D}$	Low-rank attribute matrix
$\mathbf{A} \in R^{N\times N}$	Original adjacency matrix

(Continued)

Table 1 (continued)

Notation	Meaning
$\overline{\mathbf{A}} \in R^{N\times N}$	Normalized adjacent matrix
$\mathbf{I} \in R^{N\times N}$	Identity matrix
$\mathbf{D} \in R^{N\times N}$	Degree matrix
$\mathbf{Z} \in R^{N\times d}$	Graph embedding
$\mathbf{Q} \in R^{N\times K}$	Probability Distribution

3.2 Batch Low-Rank Singular Value Decomposition Algorithm

Recent studies have demonstrated the significant effectiveness of Singular Value Decomposition in handling sparse matrices and dimensionality reduction. Inspired by their success, we introduce the Singular Value Decomposition algorithm, treating attribute information as an independent preprocessing step before training. This approach allows for the effective extraction of latent and essential information from the data while filtering out noise present in the attributes. The method is as follows:

Specifically, given a matrix $\mathbf{A} \in \mathrm{R}^{m\times n}$, low-rank Singular Value Decomposition decomposes it into the product of three matrices, where m and n can be any integers:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{T} \tag{1}$$

Here, $\mathbf{U} \in \mathrm{R}^{m\times r}$ is the left singular vector matrix, containing the structural information of the original data, where r represents the rank of the low-rank. $\mathbf{S} \in \mathrm{R}^{m\times r}$ is the diagonal matrix containing singular values, typically arranged in descending order, representing the importance of the data. $\mathbf{V}^{T} \in \mathrm{R}^{r\times n}$ is the right singular vector matrix, containing the feature information of the original data.

Singular Value Decomposition can be applied to decompose any matrix, making it adaptable to the characteristics of current attribute information. We will rewrite Eq. (1) as follows:

$$\mathbf{X}_{\mathbf{r}} = \mathbf{U}_{r}\mathbf{S}_{r}\mathbf{V}_{r}^{T} \tag{2}$$

Here, $\mathbf{X}_{\mathbf{r}} \in \mathrm{R}^{N\times D}$ represents the low-rank attribute matrix, $\mathbf{U}_{r}$ is the low-rank left singular vector matrix, $\mathbf{S}_{r}$ is the low-rank diagonal matrix, and $\mathbf{V}_{r}$ is the low-rank right singular vector matrix, where $\mathbf{V}_{r}^{T}$ is the transpose of $\mathbf{V}_{r}$.

To adapt to the Singular Value Decomposition in the case of large data, we partition the dataset $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ into M batches, where each batch $batch_i$ consists of B samples, and B is a positive integer, i.e.:

$$\boldsymbol{batch_i} = \left\{\boldsymbol{x}_{i\cdot B+1}, \boldsymbol{x}_{i\cdot B+2}, \ldots, \boldsymbol{x}_{(i+1)B}\right\} \tag{3}$$

Here, B represents the number of samples included in $batch_i$.

The objective of centralization is to set the mean of the data in each batch to zero. Assuming $batch_i$ contains B samples with a mean of μ_i, centralization is applied to each batch $batch_i$, resulting in the centered batch $batch_i^{centered}$. This process can be mathematically represented as follows:

$$\boldsymbol{batch}_i^{centered} = \boldsymbol{batch}_i - \boldsymbol{\mu}_i = \left\{\boldsymbol{x}_{i\cdot B+1} - \boldsymbol{\mu}_i, \boldsymbol{x}_{i\cdot B+2} - \boldsymbol{\mu}_i, \ldots, \boldsymbol{x}_{(i+1)B} - \boldsymbol{\mu}_i\right\} \tag{4}$$

Here, $x_{i \cdot B+2}$ represents the 2-th sample in $batch_i$, and μ_i is the mean of $batch_i$. Performing Singular Value Decomposition on each $batch_i^{centered}$ yields different low-rank matrices $\mathbf{u}_i, \mathbf{s}_i, \mathbf{v}_i$:

$$\boldsymbol{u}_{i\sim i+r}, \boldsymbol{s}_{i\sim i+r}, \boldsymbol{v}_{i\sim i+r} = \boldsymbol{SVD}\left(\boldsymbol{batch}_i^{\boldsymbol{centered}}\right) \tag{5}$$

Here, SVD represents the Singular Value Decomposition algorithm.

We concatenate all low-rank matrices $\mathbf{u}_i, \mathbf{s}_i, \mathbf{v}_i$ from each $batch_i^{centered}$ to form matrices $\mathbf{U}_r, \mathbf{S}_r, \mathbf{V}_r$:

$$\begin{aligned} \mathbf{U}_r &= \{u_1, u_2, \ldots, u_{i*r}\} \\ \mathbf{S}_r &= \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{i*r}\} \\ \mathbf{V}_r &= \{v_1, v_2, \ldots, v_{i*r}\} \end{aligned} \tag{6}$$

We perform a descending order sorting on $\mathbf{S}_r$ and select the top r values. In other words, we individually sort the real numbers of singular values in descending order and choose the top r singular values. The purpose is to emphasize the significance of singular values, allowing the primary information of attribute matrix $\mathbf{X}$ to be represented by a reduced number of singular values or vectors. The aforementioned learning process can be articulated as follows:

$$\begin{aligned} \mathbf{S}_{\boldsymbol{low}} &= \boldsymbol{descending}_{\boldsymbol{sort}}\left(\mathbf{S}_r\right) \\ \mathbf{S}_{low\_r} &= TOP\left(\mathbf{S}_{low}, r\right) \end{aligned} \tag{7}$$

Next, we will obtain the corresponding vectors of $\mathbf{V}_r$ based on the indices selected indices of the singular values $\mathbf{S}_{low\_r}$ and form $\mathbf{V}_{low\_r}$ according to Eq. (8):

$$\mathbf{V}_{\boldsymbol{low\_r}} = \boldsymbol{f}\left(\boldsymbol{x}\right) = \begin{cases} \mathbf{v}_{\boldsymbol{i}}, & \mathbf{S}_{\boldsymbol{low\_r,i}}\ \boldsymbol{match}\ \mathbf{v}_{\boldsymbol{i}} \\ 0, & \boldsymbol{others} \end{cases} \tag{8}$$

Finally, we reconstruct the low-rank attribute matrix $\mathbf{X}_r$ using $\mathbf{U}_r$, $\mathbf{S}_{low\_r}$, $\mathbf{V}_{low\_r}$ according to Eq. (9).

$$\mathbf{X}_r = \mathbf{U}_r \mathbf{S}_{\boldsymbol{low\_r}} \mathbf{V}_{\boldsymbol{low\_r}}^{\boldsymbol{T}} \tag{9}$$

3.3 Pseudo-Siamese Neural Network

In this section, we embed the nodes of both the original and enhanced data into a latent space and design a pseudo-siamese neural network with an encoder that shares the same architecture but has non-shared learnable parameters.

Residual connections allow information to propagate between network layers, aiding in mitigating the vanishing gradient problem, accelerating the training process, and enhancing model performance. In this work, the representations learned by the l-th layer of GCNs can be obtained through the following convolution operation:

$$\begin{aligned} \mathbf{Z}^{(l)} &= \varnothing\left(\overline{\mathbf{A}}\mathbf{Z}^{l-1}\mathbf{W}_{\boldsymbol{v1}}^{\boldsymbol{l-1}} + \mathbf{Z}^{l-1}\right) \\ \mathbf{Z}_r^{(l)} &= \varnothing\left(\overline{\mathbf{A}}\mathbf{Z}_r^{l-1}\mathbf{W}_{v2}^{l-1} + \mathbf{Z}_r^{l-1}\right) \end{aligned} \tag{10}$$

Here, $\varnothing$ is the activation function of the fully connected layer, such as Relu [29] or the Sigmoid function. $\mathbf{Z}^{l-1}$ and $\mathbf{Z}_r^{l-1}$ represent the embeddings of two views, and $\mathbf{W}_{v1}^{l-1}$ and $\mathbf{W}_{v2}^{l-1}$ are the weight matrices corresponding to the encoder's l-th layer. Additionally, we denote $\mathbf{Z}^{(0)}$ as the original data $\mathbf{X}$ and $\mathbf{Z}_r^{(0)}$ as the enhanced data $\mathbf{X}_r$. As shown in Eq. (10), the representations $\mathbf{Z}^{l-1}$ and $\mathbf{Z}_r^{l-1}$ will traverse

the normalized adjacency matrix $\overline{\mathbf{A}}$ to obtain new representations $\mathbf{Z}^{(l)}$ and $\mathbf{Z}_r^{(l)}$. It is important to note that the input to the first layer of GCNs consists of the original data $\mathbf{X}$ and the enhanced data $\mathbf{X}_r$:

$$\mathbf{Z}^{(1)} = \varnothing\left(\overline{\mathbf{A}}\mathbf{X}\mathbf{W}_{v1}^{1}\right) \tag{11}$$

$$\mathbf{Z}_r^{(1)} = \varnothing\left(\overline{\mathbf{A}}\mathbf{X}_r\mathbf{W}_{v2}^{1}\right)$$

We denote the embeddings output by the last layer of GCNs as $\mathbf{Z}^{(L)}$ and $\mathbf{Z}_r^{L}$.

3.4 Mutual Information-Enhanced Clustering Module

To minimize redundancy in embeddings and effectively preserve more discriminative features, we initially optimize the embeddings between cross-view samples using Mean Squared Error (MSE) loss. This ensures that the learned representations are not influenced by irrelevant information, thereby guaranteeing the quality of the latent space for subsequent clustering tasks. The formula is as follows:

$$\boldsymbol{L}_{MSE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum\nolimits_{i=1}^{n}\left(\mathbf{Z}^{(L)} - \mathbf{Z}_r^{L}\right)^{2} \tag{12}$$

Here, $\mathbf{Z}^{(L)}$ and $\mathbf{Z}_r^{L}$ represent the embeddings of the last layer of the graph convolutional network.

Next, we merge the embeddings from the two views for each node as follows:

$$\mathbf{Z} = \frac{\mathbf{1}}{\mathbf{2}}\left(\mathbf{Z}^{(L)} + \mathbf{Z}_r^{L}\right) \tag{13}$$

Here, $\mathbf{Z} \in \mathrm{R}^{N\times d}$ represents the node embeddings for clustering.

Finally, by inputting $\mathbf{Z}$ into a one-layer Multi-Layer Perceptron (MLP) with a softmax activation function, we convert it into a K-dimensional clustering space, where K represents the number of clusters. The learning process can be expressed as:

$$\mathbf{Q} = \boldsymbol{softmax}\left(\mathbf{Z}\right) \tag{14}$$

where $\mathbf{Q} \in \mathrm{R}^{N\times K}$ represents the probability matrix indicating the probability of all N nodes belonging to K clusters. We can view $\mathbf{Q}$ as a probability distribution.

Following the acquisition of the clustering probability distribution $\mathbf{Q}$, we refine the data representation by focusing on learning high-confidence data, aiming to strengthen the cohesion within the clustering. Specifically, we aim to reinforce the intra-cluster cohesion by emphasizing representations that are closer to the cluster centers. Therefore, we compute the target distribution P as follows:

$$\mathbf{P}_{ij} = \frac{\mathbf{Q}_{ij}^{2}/\boldsymbol{f}_j}{\sum_{j'}\mathbf{Q}_{ij'}^{2}/\boldsymbol{f}_{j'}} \tag{15}$$

Here, $f_j = \sum_i \mathbf{Q}_{ij}$ is the soft clustering frequency. In the target distribution $\mathbf{P}$, each assignment in $\mathbf{Q}$ is squared and normalized to give higher confidence to the assignments. This leads to the following objective function:

$$\boldsymbol{L}_{KL}(\mathbf{P}||\mathbf{Q}) = \sum\nolimits_{x}\mathbf{log}\frac{\mathbf{P}(\boldsymbol{x})}{\mathbf{Q}(\boldsymbol{x})} \tag{16}$$

Minimizing the KL divergence loss between the **Q** and **P** distributions enables the GCNs module to enhance its ability to learn representations for clustering tasks. This ensures that the data representations around cluster centers become more compact.

In order to aggregate samples within the same cluster while simultaneously separating them from samples in other clusters, we choose to periodically update the top τ probability samples from different clusters. This process enhances the cohesion of positive samples within clusters and improves the discriminative ability of sample pairs, i.e.,

$$\boldsymbol{I}\left(\boldsymbol{x}_{i,j} \textbf{ is among the top } \boldsymbol{\tau} \textbf{ elements}\right) = \begin{cases} 1, \textbf{if } \boldsymbol{q}_{i,j} \textbf{ is among the top } \boldsymbol{\tau} \textbf{ elements} \\ 0, \textbf{ otherwise} \end{cases} \tag{17}$$

Here, $I(x_{i,j}\ is\ among\ the\ top\ \tau\ elements) = 1$ indicates that sample x_i is most likely to belong to cluster c_j, and the label for this sample is recorded as y^{key}. The number of key samples in each cluster is N_k, and the total number of key samples is $C \cdot N_k$, enhancing the cohesion of positive samples within clusters.

We use the obtained pseudo-labels y^{key} to constrain the model's results, i.e.,

$$\boldsymbol{L}_{BCE}(\boldsymbol{y}^{key}, \hat{\boldsymbol{y}}) = -\frac{\mathbf{1}}{\boldsymbol{C} \cdot \boldsymbol{N}_k} \sum\nolimits_{i=1}^{C \cdot N_k} \left(\boldsymbol{y}_i^{key} \textbf{log}\left(\hat{\boldsymbol{y}}_i\right) + \left(\mathbf{1} - \boldsymbol{y}_i^{key}\right) \textbf{log}(\mathbf{1} - \hat{\boldsymbol{y}}_i)\right) \tag{18}$$

Here, y^{key} is the label of the key sample, and $\hat{y}$ is the model's clustering result.

In summary, the overall loss calculation for ECN-MF is as follows:

$$\boldsymbol{L} = \boldsymbol{\alpha} \boldsymbol{L}_{MSE} + \boldsymbol{\beta} \boldsymbol{L}_{KL} + \boldsymbol{L}_{BCE} \tag{19}$$

Here, α and β are hyperparameters.

The detailed learning procedure of ECN-MF is shown in **Algorithm 1**.

Algorithm 1: Efficient Clustering Network Based on Matrix Factorization

Input: Batch size M; Undirected graphs: $G = (\mathbf{X}, \overline{\mathbf{A}})$; Iteration number $\mathbf{I}$; Cluster number K; low-rank r; High confidence ratio τ; and updating frequency *update*.
Initialize: model parameters θ; Batches number B
for $i = 1 \rightarrow M$ **do**
  Calculate the matrix $batch_i$ for each batch using Eq. (3);
  Calculate matrix $batch_i^{centered}$ by Eq. (4);
  Calculate u_i, s_i and v_i by Eq. (5);
end
Calculate $\mathbf{U}_r$, $\mathbf{S}_r$ and $\mathbf{V}_r$ by Eq. (6);
Calculate $\mathbf{S}_{low}$, $\mathbf{S}_{low\_r}$ by Eq. (7);
Calculate $\mathbf{V}_{low\_r}$ and $\mathbf{X}_r$ by Eqs. (8) and (9), respectively;
for $j = 1 \rightarrow \mathbf{I}$ **do**
  Update embedded information $\mathbf{Z}^L$, $\mathbf{Z}_r^L$ by Eq. (10);
  Calculate L_{MSE} by Eq. (12);
  Update embedded information $\mathbf{Z}$ by Eq. (13);
  if $i \,\%\, update == 0$ **then**
    Update high confidence nodes by Eq. (17);

(Continued)

Algorithm 1 (continued)

end
Calculate $\mathbf{Q}$ and $\mathbf{P}$ by Eqs. (14) and (15), respectively;
Calculate $L_{BCE}\left(y^{key}, \hat{y}\right)$ and $L_{KL}\left(\mathbf{P}, \mathbf{Q}\right)$ by Eqs. (18) and (16), respectively;
Update the whole network by minimizing L in Eq. (19);
end
Output: The mode results $\mathbf{C}$.

4 Experiments

4.1 Dataset

We proposed ECN-MF, which was evaluated on six datasets, including CORA [12] CITESEER [12], European Air Traffic (EAT) [30], United States Air Traffic (UAT) [30], Amazon Photo (AMAP), and Amazon Computer (AMAC). Table 2 provides a brief description of these datasets.

Table 2: Summary of datasets

Dataset	Type	Sample	Edges	Dimension	Class
CORA	Graph	2708	10556	1433	7
CITESEER	Graph	3327	9928	3703	6
EAT	Graph	399	11988	203	4
UAT	Graph	1190	27198	239	4
AMAP	Graph	7650	238162	745	8
AMAC	Graph	13752	491722	767	10

4.2 Experiment Setup

All experimental results were obtained on a high-performance server equipped with an NVIDIA 3090 GPU, 64 GB RAM, and the PyTorch deep learning platform.

4.2.1 Training Procedure

Our network is trained by minimizing the loss in Eq. (19) using the Adam optimizer for 1000 iterations until convergence. After optimization, we directly apply Eqs. (13) and (14) to cluster the node embeddings of the two views and report the final convergence results for four metrics. Following all compared methods and to mitigate the adverse effects of randomness, we repeat the experiments 10 times and report the averages along with the corresponding standard deviations.

4.2.2 Parameter Settings

For the sake of fairness, regarding MCGC [14], we only executed their source code on the graph datasets listed in Table 2. For other baselines, we reproduced the results by adopting the source code with the original settings. In our proposed method, the learning rate of the optimizer is set to 1e-4 for CORA/CITESEER/EAT/AMAP/AMAC and 1e-3 for UAT. The rank r of the attribute matrix is set to 40 for CORA, 50 for CITESEER/EAT, 95 for UAT, 20 for AMAP, and 70 for AMAC. Our

encoder consists of two layers of linear MLPs, with dimensions set to 900 for CITESEER, 600 for CORA/EAT/UAT/AMAC, and 800 for AMAP.

4.2.3 Metrics

To validate the superiority of our ECN-MF compared to the baselines, we employed four widely used metrics to evaluate clustering performance, namely Accuracy (ACC), Normalized Mutual Information (NMI), Average Rand Index (ARI), and macro F1-score (F1) [31–33].

4.3 Performance Comparison

To demonstrate the superiority of our proposed ECN-MF, we compared ECN-MF with 13 base- lines. Specifically, a classification method, graphMAE2 [34], is considered. Five deep clustering methods, including AE [35], DEC [36] SSGC [37], SDCN [5], and SAGSC [38], utilize autoencoders for node encoding, followed by clustering on the learned embeddings. Two hard sample mining methods, GDCL [39], and ProGCL [40], are employed. Additionally, five deep graph clustering methods for comparison: MCGC [14], MVGRL [13], AFGRL [27], AutoSSL [41] and SCDGN [42], are incorporated. These methods are designed with contrastive strategies to enhance the discriminative capability of samples.

Table 3 reports the clustering performance of all compared methods on six benchmarks. From these results, we can derive four key observations: 1) Our ECN-MF outperforms other deep clustering methods, attributed to the benefits of contrastive learning in implicitly capturing supervisory information. 2) Compared to contrastive methods, our approach demonstrates superior performance, leveraging an information-enhanced clustering guidance module to better preserve the original data information in the embedded data. It ensures better retention of information and improves the discriminative capability of sample pairs by bringing samples from the same cluster closer and pushing away those from different clusters. 3) Our method achieves the best performance on CITESEER, showcasing the effectiveness of utilizing Singular Value Decomposition to capture latent important information of samples, particularly advantageous in handling sparse matrices. 4) Favorable results on AMAP and AMAC highlight the effectiveness of our batched low-rank singular value decomposition algorithm in handling large datasets. In summary, our method outperforms most others on six datasets with four metrics, validating the effectiveness of our proposed approach in addressing unreasonable data preprocessing and handling positive and negative sample pairs.

Table 3: Clustering performance on the six datasets (average ± standard deviation). Red and blue values represent the best and second-best results, respectively. OOM indicates Out-Of-Memory during training

Method	Metric	Classification	Classical deep graph clustering					Hard sample		Contrastive deep graph clustering					Ours
		graphMAE2	AE	DEC	SSGC	SDCN	SAGSC	GDCL	ProGCL	MCGC	MVGRL	AFGRL	AutoSSL	SCDGN	
CORA	ACC	33.88±3.26	49.38±0.91	46.50±0.26	69.28±3.70	35.60±2.83	66.58±0.13	70.83±0.47	57.13±1.23	42.85±1.13	70.47±3.70	26.25±1.24	63.81±0.57	71.18±0.64	71.57±0.64
	NMI	12.68±2.65	25.65±0.65	23.54±0.34	54.32±1.92	14.28±1.91	50.80±0.17	56.60±0.36	41.02±1.34	24.11±1.00	55.57±1.54	12.36±1.54	47.62±0.45	55.27±0.59	54.75±1.21
	ARI	08.90±1.67	21.63±0.58	15.13±0.42	46.27±4.01	07.78±3.24	40.64±0.29	48.05±0.72	30.71±2.70	14.33±1.26	48.70±3.94	14.32±1.87	38.92±0.77	49.18±1.38	48.62±1.25
	F1	32.05±4.01	43.71±1.05	39.23±0.17	64.70±5.53	24.37±1.04	63.64±0.12	52.88±0.97	45.68±1.29	35.16±0.91	67.15±1.86	30.20±1.15	56.42±0.21	69.59±0.54	64.45±1.03
CITESEER	ACC	31.48±4.02	57.08±0.13	55.89±0.20	68.97±0.34	65.96±0.31	66.58±0.13	66.39±0.65	65.92±0.80	64.76±0.07	62.83±1.59	31.45±0.54	66.76±0.67	63.43±0.18	71.80±1.23
	NMI	07.80±2.77	27.64±0.08	28.34±0.30	42.81±0.20	38.71±0.32	40.42±0.09	39.52±0.38	39.59±0.39	39.11±0.06	40.69±0.93	15.17±0.47	40.67±0.84	41.50±0.32	45.07±1.15
	ARI	05.97±2.39	29.31±0.14	28.12±0.36	44.42±0.32	40.17±0.43	41.26±0.09	41.07±0.96	36.16±1.11	37.54±0.12	34.18±1.73	14.32±0.78	38.73±0.55	41.47±0.50	46.63±2.15
	F1	30.20±4.62	53.80±0.11	52.62±0.17	64.49±0.27	63.62±0.24	62.47±0.05	61.12±0.70	57.89±1.98	59.64±0.05	59.54±2.17	30.20±0.71	58.22±0.68	59.34±0.85	64.53±1.89
EAT	ACC	34.49±1.16	38.85±2.32	36.47±1.60	32.41±0.45	39.07±1.51	46.32±0.25	33.46±0.18	43.36±0.87	32.58±0.29	32.88±0.71	37.42±1.24	31.33±0.52	32.33±0.00	51.87±0.95
	NMI	06.42±0.38	06.92±2.80	04.96±1.74	04.65±0.21	08.83±2.54	26.60±0.48	13.22±0.33	23.93±0.45	07.04±0.56	11.72±1.08	11.44±1.41	07.63±0.85	05.80±0.00	24.05±1.67
	ARI	04.24±0.49	05.11±2.65	03.60±1.87	01.53±0.04	06.31±1.95	24.00±0.53	04.31±0.29	15.03±0.98	01.33±0.14	04.68±1.30	06.57±1.73	02.13±0.67	02.55±0.00	22.75±1.03
	F1	32.87±1.59	38.75±2.25	34.84±1.28	26.49±0.66	33.42±3.10	38.93±0.12	25.02±0.21	42.54±0.45	27.03±0.16	25.35±0.75	30.53±1.47	21.82±0.98	25.11±0.00	50.64±2.63
UAT	ACC	34.51±1.18	46.82±1.14	45.61±1.84	36.74±0.81	52.25±1.91	42.94±0.57	48.70±0.06	45.38±0.58	41.93±0.56	44.16±1.38	41.50±0.25	42.52±0.64	44.86±1.62	52.01±0.96
	NMI	06.43±0.37	17.18±1.60	16.63±2.39	08.04±0.18	21.61±1.26	18.30±0.16	25.10±0.01	22.04±2.23	16.64±0.41	21.53±0.94	17.33±0.54	17.86±0.22	12.90±0.45	24.73±1.24
	ARI	04.25±0.49	13.59±2.02	13.14±1.97	05.12±0.27	21.63±1.49	13.40±0.36	21.76±0.01	14.74±1.99	12.21±0.13	17.12±1.46	13.62±0.57	13.13±0.71	11.80±0.77	23.78±1.59
	F1	32.90±1.61	45.66±1.49	44.22±1.51	29.50±1.57	45.59±3.54	38.06±1.26	45.69±0.08	39.30±1.82	35.78±0.38	39.44±2.19	36.52±0.89	34.94±0.87	41.33±1.82	43.93±1.79
AMAP	ACC	34.49±1.16	48.25±0.08	47.22±0.08	60.23±0.19	53.44±0.81	57.80±0.27	43.75±0.78	51.53±0.38	OOM	41.07±3.12	75.51±0.77	54.55±0.97	70.55±0.03	76.88±0.80
	NMI	06.43±0.37	38.76±0.30	37.35±0.05	60.37±0.15	44.85±0.83	53.60±0.41	37.32±0.28	39.56±0.39		30.28±3.94	64.05±0.15	48.56±0.71	64.02±0.04	68.21±1.10
	ARI	04.25±0.49	20.80±0.47	18.59±0.04	35.99±0.47	31.21±1.23	30.90±0.03	21.57±0.51	34.18±0.89		18.77±2.34	54.45±0.48	26.87±0.34	53.02±0.03	58.98±1.38
	F1	32.87±1.59	47.87±0.20	46.71±0.12	52.79±0.01	50.66±1.49	57.63±0.06	38.37±0.29	31.97±0.44		32.88±5.50	69.99±0.34	54.47±0.83	63.93±0.04	71.58±0.31
AMAC	ACC	35.76±1.30	21.67±0.51	19.03±0	52.22±0.01	43.04±4.09	43.89±0.01	OOM	OOM	OOM	OOM	OOM	OOM	51.64±0.04	54.12±0.72
	NMI	05.74±0.65	03.21±0.79	02.82±0	52.55±0.02	23.30±6.60	38.92±0.01							40.90±0.01	35.59±0.98
	ARI	04.18±0.75	−2.06±0.96	−1.26±0	32.30±0.03	17.40±7.49	16.83±0.02							29.69±0.04	33.21±1.25
	F1	32.69±0.83	12.16±0.78	10.84±0	39.78±0.01	20.95±3.70	45.29±0.04							51.12±0.02	32.99±0.49

4.4 Time Complexity Analysis

Firstly, we perform Singular Value Decomposition on attributes, and the time complexity of calculating the low-rank process to obtain the reconstructed attribute $\mathbf{X}_r$ in Fig. 2 is $O(batch \times BD \times r)$, where r is the rank size, B is the batch size, and D is the dimension. Subsequently, during the training phase, we employ a two-layer residual graph convolutional neural network (GCN) model. The time complexity for computing the two views is $O(2NDd + 4Ndd)$, where d is the dimension of the graph convolutional neural network, N is the number of samples, and D is the dimension of the original samples. The time complexity for calculating the mean square error loss function is $O(Nd)$, and N is the number of samples. The time complexity for high confidence selection is $O(NK)$, where K is the number of clusters, and N is the number of samples. The time complexity for computing the target distribution is $O(NK)$, where N is the number of samples. The time complexity for the KL divergence loss function is $O(NK)$, and the time complexity for the cross-entropy loss function is $O(NK)$, where N is the number of samples. Therefore, the overall time complexity of our algorithm is $O(batch \times BDr + NDd + Ndd + NK + Nd)$.

4.5 Ablation Studies

In this section, we first experimentally validate the effectiveness of our proposed data augmentation method and periodic update strategy, as shown in Table 4. For simplicity, we denote the Batch low-rank Singular Value Decomposition as B and the periodic update as P. Note that in order to replace the B operation, we use a mask to generate different views of the same node, with a mask rate of 0.5. "(w/o)B & P" represents not using the batch low-rank Singular Value Decomposition operation and periodic update, while B + P indicates the usage of both. Table 4 displays the convergence results after running 1000 epochs. Based on the observed results, we conclude that the performance would degrade without B and P, indicating that these two strategies contribute significantly to the performance improvement.

Table 4: The ablation study of the proposed Batch Low-Rank Singular Value Decomposition Operation (B) and periodic update (P) on the five datasets

		CORA	CITESEER	EAT	UAT	AMAP
ACC	(w/o)B & P	50.36	58.97	32.83	44.28	30.95
	B + P	**72.08**	**71.89**	**52.13**	**52.35**	**76.03**
NMI	(w/o)B & P	37.54	32.60	08.05	17.82	15.99
	B + P	**54.67**	**45.86**	**25.66**	**25.06**	**64.63**
ARI	(w/o)B & P	24.12	28.84	02.93	13.65	09.08
	B + P	**49.17**	**47.83**	**22.17**	**21.10**	**55.95**
F1	(w/o)B & P	52.40	55.31	26.13	40.87	18.28
	B + P	**64.33**	**64.48**	**52.05**	**44.17**	**67.67**

In Fig. 3, we visualize the accuracy throughout the entire training process until convergence using a line chart. The graph demonstrates that our model exhibits robust performance. Overall, the experimental results validate the effectiveness of B and P.

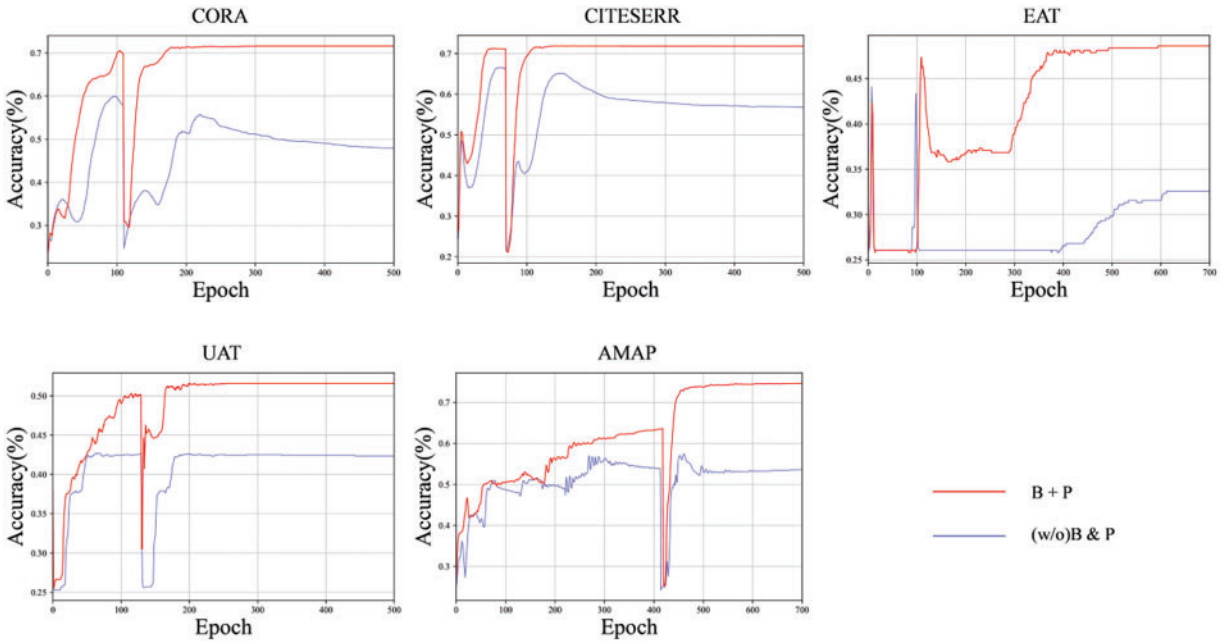


Figure 3: Accuracy analysis of the proposed batched low-rank Singular Value Decomposition (B) and periodic update (P) on the five datasets is presented. The red line represents the use of batched low-rank Singular Value Decomposition operation and periodic updates, while the blue line represents the absence of batched low-rank Singular Value Decomposition operation and periodic updates. This comparative experiment demonstrates the effectiveness of our proposed method

4.6 Hyperparameter Analysis

In this section, we will analyze the hyperparameters r, α, and β to demonstrate their impact on the dataset. For the rank r, we set the range of values from 10 to 100, and for α and β, we set the range of values from 0.1 to 1.

4.6.1 Analysis of Hyperparameter r

Fig. 4 depicts the performance variation of ECN-MF across the range of r from 10 to 100. Key observations include: 1) Appropriately setting the hyperparameter r can effectively enhance clustering performance. Without adjusting other data settings and only modifying the rank, the model achieves the highest accuracy of 73.13% on CITESEER, 73.61% on CORA, 54.39% on EAT, and 52.94% on UAT. 2) The performance of the hyperparameter r remains relatively stable over a wide range, particularly excelling on sparse datasets such as CITESEER and UAT. 3) Examining the trend in average accuracy, we observe fluctuations and declines in clustering accuracy as the rank increases, particularly on datasets like CORA and EAT. This fluctuation is attributed to the diverse characteristics of the datasets; as the rank increases, the low-rank operation might extract more low-correlation information. 4) ECN-MF requires setting an appropriate rank based on the specific features of the dataset. Nonetheless, the experimental results indicate that the range of rank values can be relatively small, reducing the model's time complexity even with a smaller rank.

Figure 4: The trend chart of clustering accuracy with varying r on four datasets demonstrates that appropriately setting a low rank reduces the model's time complexity while effectively enhancing clustering performance

4.6.2 Analysis of Hyperparameters α and β

Fig. 5 illustrates the variation in clustering performance of ECN-MF on the CITESEER dataset across the range of α and β from 0.1 to 1. When modifying only α without adjusting the other parameters, the model achieves a maximum accuracy of 72.25% on CITESEER. Similarly, when modifying only β without adjusting the other parameters, the model achieves a maximum accuracy of 72.22% on CITESEER. It can be observed from the graph that our model is not sensitive to the values of α and β, and both can yield good results.

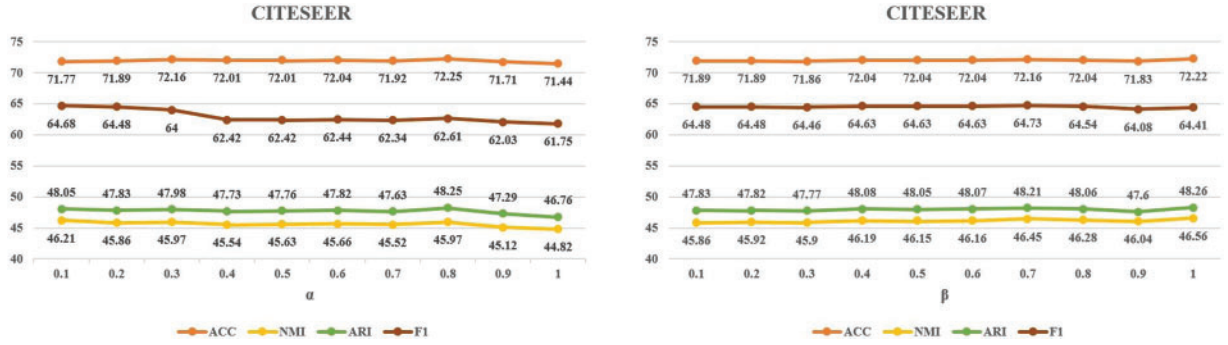


Figure 5: The trend chart of clustering accuracy with varying α and β on the CITESEER dataset, where the horizontal axis represents the values of α and β, and the vertical axis represents the values of the corresponding four evaluation metrics

4.7 Visualization Analysis

To visually showcase the superiority of ECN-MF, we employ the t-SNE algorithm (Maaten and Hinton 2008) to visualize the distribution of the learned clustering embeddings Z in a two-dimensional space. As shown in Fig. 6, ECN-MF can better reveal the intrinsic clustering structure among the data.

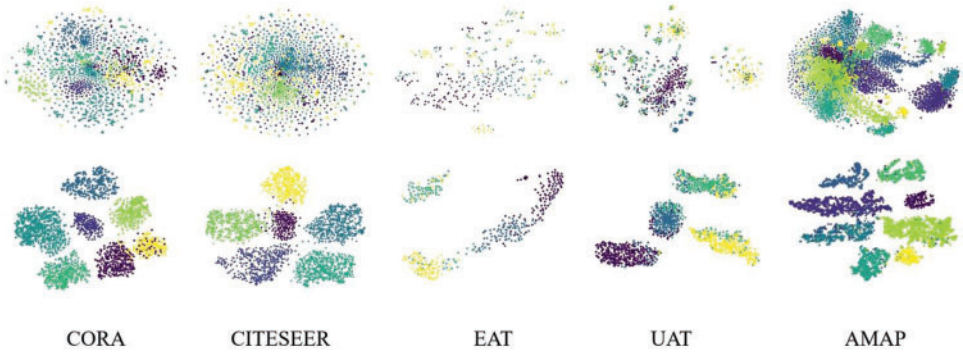


Figure 6: 2D visualization of the five datasets. The first row corresponds to the original data, while the second row corresponds to the distribution of ECN-MF. The visualization of partial samples reflects the effectiveness of our method

5 Conclusion

This paper introduces an Efficient Clustering Network based on Matrix Factorization (ECN-MF) to alleviate the negative impact of inappropriate data augmentation and enhance the quality of positive samples. By simplifying the network structure, introducing novel data augmentation methods, and designing a mutual information-enhanced clustering guidance module, ECN-MF improves its capability to handle sparse and large datasets. It brings samples from the same cluster closer while pushing samples from different clusters apart. The results of this study demonstrate the effectiveness and superiority of ECN-MF in addressing the challenges of preprocessing deep graph clustering tasks and handling positive and negative sample pairs. The paper uses hyperparameters to define the rank of the Singular Value Decomposition without special treatment of hard samples. In the future, we hope to explore new avenues of research, including: 1) employing adaptive rank selection for data, accommodating a broader range of datasets; 2) focusing more on challenging samples to enhance data mining and processing capabilities; 3) optimizing loss functions tailored for Singular Value Decomposition and challenging samples to improve clustering performance.

Acknowledgement: The authors would like to acknowledge the valuable feedback provided by the reviewers.

Funding Statement: This work was supported by the Key Research and Development Program of Hainan Province (Grant Nos. ZDYF2023GXJS163, ZDYF2024GXJS014), National Natural Science Foundation of China (NSFC) (Grant Nos. 62162022, 62162024), the Major Science and Technology Project of Hainan Province (Grant No. ZDKJ2020012), Hainan Provincial Natural Science Foundation of China (Grant No. 620MS021), Youth Foundation Project of Hainan Natural Science Foundation (621QN211), Innovative Research Project for Graduate Students in Hainan Province (Grant Nos. Qhys2023-96, Qhys2023-95).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: J. Li; analysis and interpretation of results: J. Li, F. Zeng; draft manuscript preparation: J. Cheng, J. Li, F. Zeng; data collection: Z. Tao, Y. Yang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: After the publication of the paper, the code will be made public on the author's GitHub.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Berahmand, M. Mohammadi, R. Sheikhpour, Y. Li, and Y. Xu, "WSNMF: Weighted symmetric nonnegative matrix factorization for attributed graph clustering," *Neurocomputing*, vol. 566, no. 2, pp. 127041, 2024. doi: 10.1016/j.neucom.2023.127041.

[2] K. Berahmand, Y. Li, and Y. Xu, "A deep semi-supervised community detection based on point-wise mutual information," *IEEE Trans. Comput. Soc. Syst.*, pp. 1–13, 2023.

[3] E. Pan and Z. Kang, "Multi-view contrastive graph clustering," *Adv. Neural Inf. Process Syst.*, vol. 34, pp. 2148–2159, 2021.

[4] C. Wang, S. Pan, R. Hu, G. D. Long, J. Jiang and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. Twenty-Eighth Int. Joint Conf. Artif. Intell.*, Macao, China, 2019, pp. 3670–3676.

[5] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu and P. Cui, "Structural deep clustering network," in *Proc. web conf. 2020*, Taipei, Taiwan, 2020, pp. 1400–1410.

[6] W. Tu *et al.*, "Deep fusion clustering network," in *Proc. AAAI Conf. Artif. Intell.*, Vancouver, Canada, 2021, vol. 35, pp. 9978–9987. doi: 10.1609/aaai.v35i11.17198.

[7] Z. Peng, H. Liu, Y. Jia, and J. Hou, "Attention-driven graph clustering network," in *Proc. 29th ACM Int. Conf. Multimed.*, Sichuan, China, 2021, pp. 935–943.

[8] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vision*, Paris, France, 2019, pp. 6519–6528.

[9] J. Cheng, Q. Wang, Z. Tao, D. Xie, and Q. Gao, "Multi-view attribute graph convolution networks for clustering," in *Proc. Twenty-Ninth Int. Conf. Int. Joint Conf. Artif. Int.*, Yokohama, Japan, 2021, pp. 2973–2979.

[10] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2475–2487, 2019. doi: 10.1109/TCYB.2019.2932096.

[11] Z. Tao, H. Liu, J. Li, Z. Wang, and Y. Fu, "Adversarial graph embedding for ensemble clustering," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, 2019, pp. 3562–3568.

[12] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, California, USA, 2020, pp. 976–985.

[13] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Int. Conf. Mach. Learn.*, Vienna, Austria, 2020, pp. 4116–4126.

[14] W. Xia, S. Wang, M. Yang, Q. Gao, J. Han and X. Gao, "Multi-view graph embedding clustering network: Joint self-supervision and block diagonal representation," *Neural Netw.*, vol. 145, no. 11, pp. 1–9, 2022. doi: 10.1016/j.neunet.2021.10.006.

[15] Y. Liu *et al.*, "Deep graph clustering via dual correlation reduction," in *Proc. AAAI Con. Artif. Intell.*, Vancouver, Canada, 2022, vol. 36, pp. 7603–7611.

[16] L. Liu, Z. Kang, J. Ruan, and X. He, "Multilayer graph contrastive clustering network," *Inf. Sci.*, vol. 613, pp. 256–267, 2022. doi: 10.1016/j.ins.2022.09.042.

[17] Y. Yang and H. Wang, "Multi-view clustering: A survey, big data mining and analytics," *Big Data Min. Anal.*, vol. 1, no. 2, pp. 83–107, 2018.

[18] X. Cai, C. Huang, L. Xia, and X. Ren, "LightGCL: Simple yet effective graph contrastive learning for recommendation," in *Eleventh Int. Conf. Learn. Rep.*, 2022, pp. 1–15.

[19] G. A. Khan, J. Hu, T. Li, B. Diallo, and H. Wang, "Multi-view data clustering via non-negative matrix factorization with manifold regularization," *Int. J. Mach. Learn. Cybern.*, vol. 13, no. 3, pp. 1–13, 2022. doi: 10.1007/s13042-021-01307-7.
[20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Int. Conf. Mach. Learn.*, Vienna, Austria, 2020, pp. 1597–1607.
[21] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *Int. Conf. Mach. Learn.*, 2021, pp. 12310–12320.
[22] H. Zhong *et al.*, "Graph contrastive clustering," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9224–9233.
[23] X. Yang, X. Hu, S. Zhou, X. Liu, and E. Zhu, "Interpolation-based contrastive learning for few-label semi-supervised learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 2054–2065, 2022. doi: 10.1109/TNNLS.2022.3186512.
[24] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang and Y. Shen, "Graph contrastive learning with augmentations," *Adv. Neural Inf. Process Syst.*, vol. 33, pp. 5812–5823, 2020.
[25] J. Xia, L. Wu, J. Chen, B. Hu, and S. Z. Li, "SimGRACE: A simple framework for graph contrastive learning without data augmentation," in *Proc. ACM Web Conf. 2022*, Barcelona, Spain, 2022, pp. 1070–1079.
[26] P. Bielak, T. Kajdanowicz, and N. V. Chawla, "Graph barlow twins: A self-supervised representation learning framework for graphs," *Knowl. Based Syst.*, vol. 256, no. 4, pp. 109631, 2022. doi: 10.1016/j.knosys.2022.109631.
[27] N. Lee, J. Lee, and C. Park, "Augmentation-free self-supervised learning on graphs," *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 7, pp. 7372–7380, 2022. doi: 10.1609/aaai.v36i7.20700.
[28] L. Gong, S. Zhou, W. Tu, and X. Liu, "Attributed graph clustering with dual redundancy reduction," in *IJCAI*, 2022, pp. 1–7.
[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML-10)*, Haifa, Israel, 2010, pp. 807–814.
[30] N. Mrabah, M. Bouguessa, M. F. Touati, and R. Ksantini, "Rethinking graph auto-encoder models for attributed graph clustering," *IEEE Trans. Knowl. Data Eng.*, pp. 1–31, 2022.
[31] S. Zhou, *et al.*, "Multiple kernel clustering with neighbor-kernel subspace segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1351–1362, 2019. doi: 10.1109/TNNLS.2019.2919900.
[32] S. Wang *et al.*, "Fast parameter-free multi-view subspace clustering with consensus anchor guidance," *IEEE Trans. Image Process.*, vol. 31, no. 4, pp. 556–568, 2021. doi: 10.1109/TIP.2021.3131941.
[33] S. Wang, X. Liu, L. Liu, S. Zhou, and E. Zhu, "Late fusion multiple kernel clustering with proxy graph refinement," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 4359–4370, 2021.
[34] Z. Hou *et al.*, "GraphMAE2: A decoding-enhanced masked self-supervised graph learner," in *Proc. ACM Web Conf. 2023*, Austin, Texas, USA, 2023, pp. 737–746.
[35] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *Int. Conf. Mach. Learn.*, Sydney, Australia, 2017, pp. 3861–3870.
[36] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Int. Conf. Mach. Learn.*, New York, USA, 2016, pp. 478–487.
[37] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *Int. Conf. Learn. Rep.*, 2020, pp. 1–15.
[38] C. Fettal, L. Labiod, and M. Nadif, "Scalable attributed-graph subspace clustering," *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 6, pp. 1–9, 2023. doi: 10.1609/aaai.v37i6.25918.
[39] H. Zhao, X. Yang, Z. Wang, E. Yang, and C. Deng, "Graph debiased contrastive learning with joint representation clustering," in *Proc. Thirtieth Int. Joint Conf. Artif. Intell.*, 2021, pp. 3434–3440.
[40] J. Xia, L. Wu, G. Wang, J. Chen, and S. Li, "ProGCL: Rethinking hard negative mining in graph contrastive learning," in *Int. Conf. Mach. Learn.*, Maryland, USA, 2022, pp. 24332–24346.
[41] W. Jin, X. Liu, X. Zhao, Y. Ma, N. Shah and J. Tang, "Automated self-supervised learning for graphs," in *10th Int. Conf. Learn. Rep. (ICLR 2022)*, 2022.
[42] Y. Ma and K. Zhan, "Self-contrastive graph diffusion network," in *Proc. 31st ACM Int. Conf. Multimed.*, Ottawa, Canada, 2023, pp. 3857–3865.