DeepEDD: Nonparametric Deep Graph Clustering Through Effective Distance and Density

Yuanchi Ma[a], Hui He[b], Zhongxiang Lei[a], Kaize Shi[c], Xueping Peng[c] and Zhendong Niu[a,*]

[a]Beijing Institute of Technology, The School of Computer Science and Technology, Beijing, 100081, China
[b]Beijing Institute of Technology, The School of Medical Technology, Beijing, 100081, China
[c]University of Technology Sydney, The School of Computer Science, Sydney, NSW 2007, Australia

ARTICLE INFO

Keywords:
Deep Clustering
Number of clusters
Graph Learning

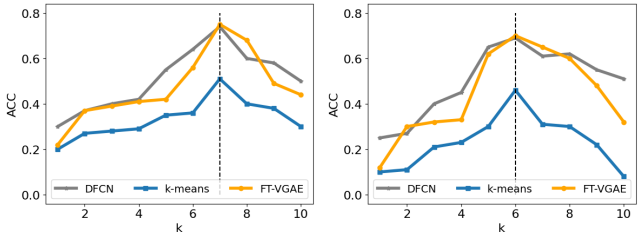
ABSTRACT

Deep graph clustering methods have become a prominent area of research. Although nonparametric clustering techniques have clear advantages, current research indicates that nonparametric methods in the field of graph clustering are scarce. Most deep graph clustering methods are parametric and require the number of clusters k to be specified a priori. An unknown k significantly diminishes the clustering performance of the model, and alternative criteria for judgment come with a computational cost. This issue is particularly pronounced in the context of deep learning, due to the increase in the number of training sessions. To address this issue, we propose an end-to-end framework, namely, nonparametric Deep graph clustering via Effective Distance and Density (DeepEDD), which utilizes the data from the clustering process to predict the k without additional computational cost. Specifically, DeepEDD introduces a new clustering algorithm based on effective distance and density as a second decoder on top of the graph masking-based fusion autoencoder framework. The algorithm does not need to define k in advance because it can automatically determine k by polynomial curve fitting. We also design a novel fusion loss to optimize DeepEDD. Experimental results on six publicly available datasets demonstrate that our proposed method surpasses existing nonparametric methods, both classical and deep. Moreover, the mask-based approach enhances the generalization performance of our framework.

1. Introduction

Clustering is a critical task in unsupervised learning, where class labels are unavailable. In an unsupervised (and more realistic) setting, this task becomes even more challenging as the number of clusters, k, is unknown. The underlying principle of deep learning is to enable computers to construct complex concepts from simpler ones [1], an ability that makes clustering of large, high-dimensional data sets efficient. Therefore, determining the true number of clusters, k, is of paramount importance. An erroneous choice of k during the iterative training process can result in significant time and energy costs, ultimately leading to negative social and environmental consequences [2].

Nonparametric methods (which automatically determine k) offer significant advantages over parametric methods (which require a predefined k), mainly due to the inherent uncertainty surrounding the correct number of clusters in real-world scenarios. Additionally, the automatic determination of the optimal k by nonparametric methods enhances clustering performance. 1) Dividing a single cluster into multiple clusters alters the clustering labels, potentially leading to convergence toward better local optima and improving performance [3]. 2) Achieving accurate clustering results depends on identifying the correct k. As shown in Fig.1, experimental results from several state-of-the-art (SOTA) parametric methods, including FT-VGAE [4], demonstrate that an incorrect choice of k can have significant negative consequences.



(a) Cora (b) Citeseer

Figure 1: The baseline on the Cora and Citeseer dataset includes the current SOTA method based on the trend of ACC changes in the number of clusters k. Where the dashed line represents the true k.

Despite the practical advantages of nonparametric methods, whether applied to conventional clustering tasks on general datasets [5] or more complex graph-structured data [6, 7], parametric methods still dominate in terms of superior performance. Nonparametric methods, such as those in Ronen et al. [2] and Zhao et al. [8], remain scarce and, unfortunately, are neither scalable nor efficient enough. Similarly, traditional clustering methods that use unsupervised criteria to determine the number of clusters [9, 10] cannot be seamlessly applied to graph data or deep clustering. To address this gap, we propose a novel deep nonparametric graph clustering approach, DeepEDD. Remarkably, even when the value of k is predefined, DeepEDD achieves

E-mail: zniu@bit.edu.cn
*Corresponding author
ORCID(s):

performance comparable to parametric methods, despite the potential inherent advantages afforded to parametric methods, as demonstrated in Table 4.

The density peak clustering algorithm, CFSFDP [11], serves as a key inspiration [12] for our approach, enabling the use of potential density data from the clustering process to determine k without incurring additional computational costs. Presently, there is a lack of awareness regarding any work that integrates this algorithm with deep learning methodologies for graph data applications. This is due to the algorithm's inherent limitations in addressing graph topology and its reliance on human judgment for threshold determination and the number of clusters. Based on this approach, we transformed the topological structure of the graph into a learnable effective distance and proposed a novel algorithm—Effective Distance and Density (EDD). EDD effectively maps the topological relationships among nodes into an inner product space, enabling more intuitive and discriminative distance representations. Furthermore, it introduces an entropy-based and polynomial-fitting strategy to automatically determine the optimal number of clusters, k, thereby overcoming the limitations of the original CFSFDP algorithm, which relies heavily on manual parameter tuning. This improvement not only enhances the accuracy of k-selection but also ensures comprehensive utilization of the graph's structural information during clustering. All the parts related to EDD have been designed by us as an independent decoder and integrated into the overall framework. After learning graph embeddings through a graph masking-based fusion autoencoder, we perform multi-target decoding. This approach mitigates the overfitting and excessive emphasis on proximity information inherent to the graph autoencoder structure [13], while also improving the model's generalization capability. Although feature learning is not our primary focus, we provide relevant experimental results. A novel loss function is designed to enhance the quality of the embeddings, working in tandem with a self-optimization module for further refinement. Our key contributions include:

- A nonparametric deep graph clustering algorithm is proposed, where we improve a new algorithm based on effective distance and density, EDD.
- A new loss is designed for optimizing our model, in addition to which we prove the convergence.
- To the best of our knowledge this is also the first work to apply a graph masking autoencoder to the prediction of the number of clusters. Specifically, the integration of the mask graph mechanism endows our model with enhanced generalization capabilities.
- Extensive experiments on six datasets demonstrate that our model outperforms state-of-the-art baselines.

The rest of this paper is as follows: § 2 presents a literature review. § 3 introduces common symbols for the method part. § 4 is devoted to our DeepEDD algorithm. The relevant theoretical derivation and proof are presented in § 5. The experiments are conducted in § 6 to support our algorithmic model and ideas. The conclusions and potential future work are discussed in § 7.

2. Related Work

2.1. Nonparametric Deep Clustering Methods

Among the limited nonparametric methods [2, 8, 14, 15, 16] designed to predict the number of clusters, we regard DeepDPM [2] and DED [16] as the most representative. These methods exemplify two approaches: two-step deep clustering and end-to-end deep clustering, respectively. DED capitalizes on feature representation learning and density-based clustering algorithms. It uses a deep convolutional autoencoder [17] and t-SNE to extract appropriate embedding features, followed by the application of a novel density peak clustering algorithm [11] to predict the number of clusters. In contrast, DeepDPM, which is based on the Dirichlet process, dynamically adjusts its architecture to estimate the evolving k using a split/merge framework. Although it has achieved state-of-the-art performance on the ImageNet dataset, its inability to capture graph topology information and its limitations in interpretability and generalization present notable challenges.

2.2. Deep Graph Clustering

In deep graph clustering, research predominantly focuses on single-structure or fused graph autoencoders. For primarily single graphical neural networks, such as GCN or GAT structures, examples include works by Kipf and Welling [6], Wang et al. [18], and Zhang and Li [19]. However, critics argue that these structures may cause the model to overemphasize graph topology, leading to the neglect of node feature information. To mitigate this, a fusion graph autoencoder structure has been introduced. The core idea is to combine a linear neural network layer with the coding layer of the graph autoencoder, producing a more robust fused representation. For instance, Bo et al. [20] applies linear fusion between layers, while Tu et al. [21] employs a deep fusion mechanism with triple validation. Compared to our enhanced masked graph autoencoder, as discussed in [22, 23], these GAE-based methods disproportionately emphasize structural, proximity, and feature information, which significantly limits their capacity to learn effective graph embeddings.

Recently, contrastive learning-based deep graph clustering methods have garnered significant attention [1, 24, 25, 26, 27]. These algorithms employ contrastive learning to extract effective feature representations by modeling the similarities and differences between samples. Central to this approach is the use of positive and negative sample pairs to train neural networks, encouraging the model to cluster similar data points closer together while pushing dissimilar points further apart. For example, Liu et al. [28] utilizes a simple low-pass denoising operation as a standalone preprocessing step for neighbor information aggregation, with only two multi-layer perceptrons serving as the primary components for contrastive learning. Similarly, Pan and

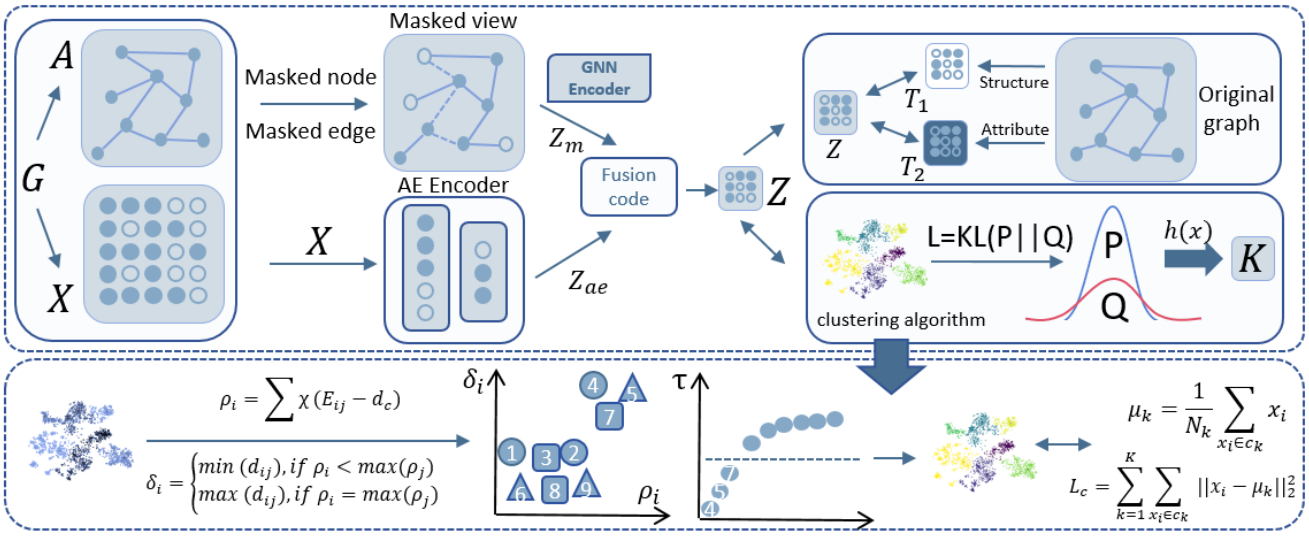


Figure 2: Illustration of DeepEDD. The top half represents the overall model architecture. Given an input graph G, output the clustering results and the number of clusters k. The bottom half represents our proposed new algorithm via effective distance density, EDD.

Kang [29] learns a consistent graph by comparing it with a loss-regularized filter to remove high-frequency noise. While these algorithms often deliver high performance, their reliance on complex data augmentation and time-intensive graph convolution operations for graph data diminishes their overall efficiency. More critically, the aforementioned deep graph clustering methods are parametric in nature, requiring the correct value of k to be predefined.

It is noteworthy that most existing nonparametric graph clustering methods are refinements of traditional algorithms, such as those derived from affinity propagation [30, 31] or Symmetric Nonnegative Matrix Factorization [32, 33]. Additionally, our literature review indicates that there have been limited recent advancements in non-parametric deep graph clustering over the past two years, such as RGC [34] and GMpool [35]. However, GMpool is a non-parametric method specifically developed for graph pooling. Overall, these nonparametric methods exhibit performance limitations when applied to generalized graph clustering.

3. Notations

This work mainly uses graph data. The graph is represented as $G = (V, E, X, A)$, where the set of nodes is denoted as $V = \{v_1, v_2, ..., v_N\}$, where v_i represents the i-th node. The set of edges between nodes is denoted as $E = \{e_{ij}\}$, and the edge between node i and node j is denoted as e_{ij}. The topology of the graph G can be represented by the adjacency matrix A, where $A_{ij} = 1, e_{ij} \in E$; otherwise, $A_{ij} = 0$. $\boldsymbol{X} = \{x_1, x_2, ..., x_N\}$ is the node v_i eigenvector attribute value, where $x_i \in \mathbb{R}^{N \times d}$. The notations commonly used in § 4 are summarized in Table 1.

Table 1
Commonly used notations

Notations	Descriptions
$G = (V, E, X, A)$	A graph G consisting of a node set V, edge set E, feature matrix X and an adjacency matrix A
v_i	A node $v_i \in V$
e_{ij}	An edge $e_{ij} \in E$
A_{ij}	A link $A_{ij} \in A$
x_i	A node feature $x_i \in X$
k	The number of clusters
N	The number of nodes with $N = \|V\|$
d	The dimension of nodes feature vector
G'	Mask graph
Z_m	Mask embedding
Z_{ae}	Hidden representation by AE network
Z	Fusion embedding
S	Mask Embedded compressed representation after re-masking
D_n	A collection of mapping functions for reconstruction
P_n	Multi-objective embedded representation
ρ	Local density
δ	Minimum distance to a sample point with higher density
d_c	Truncation distance
d_{ij}	Euclidean distance between two sample
Ef	The effective distance between nodes
$h(x)$	Determiner of cluster centers
$f(x)$	Estimator for the number of clusters
CH	The number of clusters determined by the CH value

4. Method

The overall framework of the algorithm comprises a picture mask fusion encoder and two decoders. The algorithm follows this general flow: First, graph data is input into the encoder and processed by the masking fusion encoder. The encoder generates graph mask encoding and autoencoder (AE) encoding, which together form the mixed embedding. This mixed embedding is then fed into multiple reconstruction and prediction modules. Finally, a self-optimization module refines and optimizes all loss functions in the training model, thereby completing the overall training process.

We present the entire framework in the sequence of data input and output. Section 4.1 introduces a novel mask-based

fusion coding network. Section 4.2 explains the principles of the multi-objective decoder and introduces a new loss function. Section 4.3 provides details on the new density-based clustering algorithm and its integration into the overall framework. Section 4.4 describes the self-optimization module, while Section 4.5 defines the node loss functions for the entire framework. The complete process is illustrated in Figure 2.

4.1. Masking Fusion Encoder

The objective of mask autoencoder is to reconstruct masked signals from unmasked inputs within an autoencoder framework, and recent advances in both language and vision research have provided valuable insights. Notable examples include BERT [36] and MAE [37]. In fact, mask autoencoder is also well-suited for graph data, as each edge can easily be masked or unmasked in a self-supervised manner. The core concept is to remove parts of the input graph and learn to predict the removed components, such as edges and nodes. The advantage of masks lies not only in their elimination of costly annotation but also in their ability to allow GAE to learn improved representations to some extent. However, most existing work [38, 39, 40] remains focused at the feature learning level, with limited exploration of graph clustering tasks. Furthermore, masked graph autoencoders tend to prioritize learning graph topological information.

To overcome these limitations, we propose the masking fusion encoder, which incorporates specific improvements for the graph clustering task. Specifically, we use a graph masking neural network and an autoencoder to learn the topology information and node information respectively, and adjust the learning measures of the two to achieve cross-modal information learning balance.

The initial step involves the graph masking and encoding process. The original graph G serves as the input and is masked into $G' = \text{mask}(G)$ through edge masking and feature masking. The encoder can employ architectures such as GAT or GCN. It is important to note that we simultaneously mask both types of modal information to enhance the model's cross-modal learning capability, although this also presents challenges for reconstruction. As a result, the multi-objective reconstruction method we propose differs from Li et al. [38], as we calculate the loss based on mutual information [41] and cross-entropy [6].

Formally, we represent the mask process as M, where $M = \{M_e, M_f\}$. Here, the edge and feature mask matrix is denoted as $M_e \in \{0,1\}^{|V|\times|V|}$ and $M_f \in \{0,1\}^{|V|}$, both being randomly generated binary matrices. The degree of perturbation can be controlled by the sparsity of M_e and M_f. Consequently, we obtain $M(G = \{V, A, X\}) = \{V, A * M_e, X\text{diag}(M_f)\} = \{V, A', X'\} = G'$, where A', X', and $*$ respectively represent the masked adjacency matrix, the mask feature matrix, and the Hadamard product. The nodes that are masked with edge or node characteristics are referred to as $\widetilde{V}$. The mask graph is then input to the encoder to obtain the potential representation Z_m.

We introduce an AE network for extracting node feature information to alleviate the problem of overemphasis on topological information in graph neural networks. Each layer of the AE computes the node features as follows, where w^l and b^l represent weight and bias.

$$Z_{ae}^l = f(w_l Z_{ae}^{l-1} + b^l) \tag{1}$$

We utilize a linear fusion mechanism to integrate the two graph embeddings, producing a more comprehensive and robust representation, as shown in Eq.(2). The learnable coefficient, ϵ, is optimized using the gradient fitting method, enabling fine-tuning of the relative importance of the two embedded components. In this study, ϵ is initialized to 0.1.

$$Z = (1-\epsilon)Z_m + \epsilon Z_{ae} \tag{2}$$

4.2. Multi Target Decoder

Various graph models concentrate on distinct aspects of graph information. For instance, node2vec [42] primarily captures graph structural information, while PCA, trained on the feature matrix X, predominantly encodes node attribute information. Consequently, we propose a novel loss that integrates link prediction and node reconstruction.

We employ link prediction as the target task for edge reconstruction. Instead of directly using the embedded representation of a method as the reconstruction target, we first perform element-wise multiplication on the hidden node features to generate their cross-representation. Element-wise multiplication emphasizes shared attributes and differential information, retaining only elements with high correlations between two nodes, which helps the decoder filter out inconsistent and redundant information. After obtaining the cross-correlation representation of the edge, we use an MLP layer to predict the probability of its existence. The loss function for this part, L_{p_1}, is computed following the standard form in Eq. (3), where $z_u, z_v \in Z$.

$$L_{p_1} \rightarrow h_\omega(z_u, z_v) = \text{Sigmoid}(\text{MLP}(z_u \circ z_v)) \tag{3}$$

Subsequently, leveraging the KL divergence from the GAE reconstruction framework, we balance the predicted probability of existence, construct the loss function L_{p_1}.

When reconstructing node targets, we aim to maximize the correlation between node embeddings and real data by leveraging the concept of mutual information. Therefore, we directly select AE embeddings. Given graph G as input, AE maps the feature matrix into a lower-dimensional embedding space, preserving most of the useful information and eliminating noise. Since AE has relatively low time and space complexity, it is also effective in this regard.

But before counting the loss of the node portion, re-masking [39, 43] to obtain a more compressed representation denoted as $\hat{V} \in \mathbb{R}^{|V|\times d}$. Unlike the initial masking applied to the input, the remasking occurs on the embeddings of the masked nodes, meaning that the positions of the feature matrix and feature map for these nodes remain unchanged. Node features contain greater redundancy than

links, allowing this step to learn a more compact embedded representation, thereby further reducing decoding overhead. Thus, each node v_i can be represented as $\hat{v}_i = \hat{V}[i,:] = \text{encoder}(A', X')$. We remask Z_m to obtain $S \in \mathbb{R}^{|V|\times d}$. We implement the decoder through a set of mappings represented as $\{D_n\}$, where each $D \in \{D_n\}$ is implemented by an MLP. These mapping functions are designed to map S to the target embedding space $P_n = \{p_i\} \in \mathbb{R}^{|V|\times d}, i \in [1, N]$. Each p_i serves as a target. In the actual calculation, we adopt AE structure-only embedding as the target of the reconstructed nodes. Potential p_i also include embeddings such as node2vec that contain node information.

In this case, we maximize the mutual information between $D_n(S)$ and P_n, formulating it as a specific function L_{p_2} to compute the node reconstruction loss.

To sum up, the new loss function L_p uses the KL scatter and InfoNCE as the basis for calculating the loss, and $L_p = L_{p_1} + L_{p_2}$ is represented as follows.

$$\begin{aligned} L_p = -(&\frac{1}{|\Sigma^+|}\sum_{(u,v)\in\Sigma^+} log h_w(z_u, z_v) \\ &+ \frac{1}{|\Sigma^-|}\sum_{(u',v')\in\Sigma^-} log(1 - h_w(z_{u'}, z_{v'}))) \\ &+ exp(\frac{1}{\xi}\cdot\frac{D_n(S)\cdot P_n}{||D_n(S)||\cdot||P_n||}) \end{aligned} \tag{4}$$

Where Σ^+ is the set of positive edges, Σ^- is the set of negative edges sampled from the graph, z is the encoding, and h is the decoder in link prediction. ξ is the hyperparameter. A detailed derivation is given in section § 5.

4.3. Effective Distance and Density Decoder

The density peaks clustering algorithm [11] offers an elegant solution to the issue of pre-defining k, although it was not initially developed for graph data. Its key advantages include low computational complexity and independence from the shape of the data in the underlying space. These properties make it a suitable reference point for addressing the clustering problem in graph data. It [11] defines two main criteria: the number of points surrounding a sample point, referred to as the local density, ρ, and the minimum distance to a sample point with higher density, referred to as δ. In clustering, the points near the cluster center have higher ρ, meaning the cluster center exhibits the highest local density. Additionally, data points with higher δ are far from other points with higher density.

Building on this algorithm, we propose the EDD algorithm and extend its application to graph clustering. Specifically, we first generalize the Euclidean distance between sample points in the inner product space to the effective distance between nodes in the graph structure. Secondly, the number of clusters, k, is determined through polynomial fitting, a newly developed method that enhances the accuracy of k estimation.

For a sample point z_i in the feature set Z, its local density is defined as $\rho_i = \sum_{1\le j\ne i\le N}\chi(d_{ij}, d_c)$. In the original density peaks clustering algorithm, d_{ij} represents the Euclidean distance between data points z_i and z_j, d_c is the truncation distance, and χ is the kernel function. However, this approach is not directly applicable to graph-structured data, as neglecting graph topology information results in distances between data points deviating from their true values, thereby degrading clustering performance. We propose a new method for computing d_{ij}, based on the observation that nodes with direct links are closer in graph data. Therefore, we incorporate the adjacency matrix A to adjust the actual distance between data points, which we refer to as the effective distance between nodes, denoted by Ef. The formula is as follows.

$$\begin{aligned} Ef_{ij} &= \hat{d}_{ij} + I(A_{ij})/\iota \\ \hat{d}_{ij} &= d_{ij}/\sum_{o=1}^{N} d_{io} \end{aligned} \tag{5}$$

Where $\hat{d}_{ij}$ denotes the Euclidean distance of z_i normalized in relation to the other points, I represents the inverse operation, A_{ij} represents whether there is a link between nodes i and j on the graph, and ι serves as the weight parameter governing the adjacency matrix, initialized to 1×10^{-6}. Consequently, a new ρ_i is as follows, and a Gaussian kernel is selected as the kernel function.

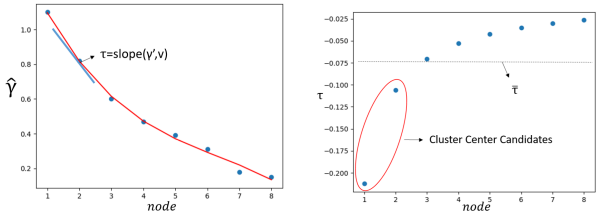
$$\rho_i = \sum_{1\le j\ne i\le N}\chi(Ef_{ij}, d_c) = \sum_{j\ne i} e^{-(Ef_{ij}/d_c)^2} \tag{6}$$

It is important to note that the optimal truncation distance, d_c, is crucial for determining accurate clustering results. According to Eq. (6), the value of d_c significantly impacts the calculation of sample point density ρ. In the original method, d_c is manually selected based on experience, which is neither stable nor automatic. To address this issue, we propose a method to automatically determine d_c using information entropy within the data field.

Consider the dataset as a data field, where the local density ρ_i of a data point corresponds to its potential energy function ϕ_i within the field, as defined in Eq. (7). In this context, the influencing factor σ represents the truncation distance d_c.

$$\begin{aligned} H &= -\sum_{i=1}^{N}\frac{\phi_i}{\sum_{i=1}^{N}\phi_i} log(\frac{\phi_i}{\sum_{i=1}^{N}\phi_i}) \\ \phi_i &= \sum_{j=1}^{N} e^{-(d_{ij}/\sigma^2)} \end{aligned} \tag{7}$$

We employ information entropy H to quantify the overall uncertainty of the dataset. Based on ϕ_i, we derive the variation curve of H, and the optimal value of d_c is identified at the point where H reaches its minimum. The underlying principle is that when H is minimized—indicating the



(a) Curve-fitting plot (b) The change in slope τ

Figure 3: Curve fitting to determine clustering center.

lowest entropy—the dataset becomes more stable, allowing for the determination of the most appropriate d_c.

Subsequently, we define the high-density nearest-neighbor distance δ_i, $\delta_i = \min_{j:\rho_j>\rho_i}(d_{ij})$, if $\rho_i < \max\{\rho_j\}$ and $\delta_i = \min_j(d_{ij})$, if $\rho_i = \max\{\rho_j\}$.

This yields the formula for determining k. We consider the product of ρ and δ for each sample point as input. In Eq. (8), $f(x)$ acts as an estimator for the number of clusters, identifying points where the product of ρ and δ is significantly larger than that of other sample points, indicating potential cluster centers. The input to $f(x)$ is $\hat{\gamma}$, and the output is the value of k closest to and less than the threshold. $h(x)$ serves as a determiner of cluster centers, referencing other clustering metrics, with a default preference for using the CH value [9] to assist in determining the number of clusters. The input of $h(x)$ is the embeddings Z and $f(\hat{\gamma})$, and the output is the final determined k value. In summary, we have the following Eq. (8) to determine the value of k.

$$k = h(f(\{\delta_i \cdot \rho_i\}), Z) = h(f(\hat{\gamma})) \tag{8}$$

To start with, we introduce the $f(x)$ clustering number estimator. We define the set of products of ρ_i and δ_i as $\hat{\gamma}_i$. These values are ordered from largest to smallest, and polynomial curve fitting is applied to compute estimates for each ρ_i and δ_i, which we denote as γ'_i, as shown in Fig.3. After sorting, we calculate the slope of the tangent line to the curve at each point, denoted as τ_i, where $\tau_i = \text{slope}(\gamma'_i, v_i)$. We then calculate the average τ_i value, $\bar{\tau}$, which is used as a threshold. Since the $\hat{\gamma}_i$ of the cluster center is significantly larger than that of the other points, we select the point where τ_i is significantly smaller than $\bar{\tau}$ as a candidate cluster center.

Subsequently, $h(x)$ serves as the clustering determiner, as single indicators often fail to comprehensively assess the clustering quality of the entire dataset. Specifically, the calculated $\hat{\gamma}_i$ emphasizes closer in-cluster distances but overlooks inter-cluster information. The CH index, calculated as the ratio of intra-cluster dispersion to inter-cluster dispersion, provides a measure to incorporate inter-cluster information effectively. By selecting the optimal cluster number based on the CH value, we aim to balance the prediction results and improve cluster number estimation. Integration experiments, as shown in Table 13, highlight the associated performance improvements. The overall formula for the $h(x)$ clustering determiner is as follows, where ν is the adaptive equilibrium parameter. After extensive experiments, the optimal value for ν was found to be 0.95.

$$h(x) = \nu f(x) + (1-\nu)CH. \tag{9}$$

We aim to minimize the spacing within the same cluster to guide the network in influencing the distribution of learning data. Consequently, we calculate the mean squared error between each sample vector and its corresponding cluster mean vector. The mean vector of sample points within each cluster is denoted as μ_k, where C_k represents cluster k and N_k denotes the number of samples contained in cluster C_k. This approach leads to the formulation of the loss function. A detailed proof of the validity of this part of the decoding is provided in section § 5.

$$L_c = \sum_{i=1}^{k} \sum_{x_i \in C_k} ||x_i - \mu_k||_2^2, \mu_k = \frac{1}{N_k} \sum_{x_i \in C_k} x_i \tag{10}$$

4.4. Self-optimizing Modules

The core idea of the self-supervised optimization graph embedding is that compels the current distribution Q to converge toward the target distribution P by minimizing the KL divergence loss between the Q and P distributions. The same approach has been used in a number of previous works [18, 19, 20], and it has been shown to optimize the learned code, and we continue to use this approach. We utilized the t-distribution of Students [44] to measure it. The similarity between the node embedding z_i and the clustering center r_u was initially measured using the q_{iu} metric. Thus our optimized affiliation matrix p is constructed based on the obtained q. To minimize the clustering loss, we construct the loss using the KL scatter.

$$q_{iu} = \frac{(1 + ||z_i - r_u||^2/v)^{-1}}{\sum_k (1 + ||z_i - r_k||^2/v)^{-1}} \tag{11}$$

Thus our optimized affiliation matrix p is constructed based on the obtained q as follows.

$$p_{iu} = \frac{q_{iu}^2 / \sum_i q_{iu}}{\sum_k (q_{ik}^2 / \sum_i q_{ik})} \tag{12}$$

We consider fusion coding as a Q-distribution. This can result in a smaller spacing between points within the same cluster and a larger spacing between clusters.

$$L_s = KL(P||Q) = \sum_i \sum_j p_{iu} log \frac{p_{iu}}{q_{iu}} \tag{13}$$

4.5. Joint Embedding

Our model is end-to-end, the aforementioned modules collaboratively optimize graph embedding and clustering learning. The total objective function is defined as follows.

$$L = L_p + \alpha L_c + \beta L_s \tag{14}$$

where α and β are used to balance the various losses. The overall algorithm is shown in Algorithm 1.

Algorithm 1 GRAPH CLUSTERING WITH MASKED AUTOENCODERS (DeepEDD)

Input: Graph G; Number of iterations $Iter$; Number of epoch $epoch$; Number of layers $Layers$.

While max epoch < $epoch$ or convergence **do**
  While max iterations < $iter$ **do**
    Randomly masking G as G';
    Encoding of data according to the encoder.
  Get Z_m, Z_{ae} and according Eq.(2) to get Z;
    Decoding of multiple reconstruction targets based on Eq. (4) to get L_p;
  Truncation distance d_c is determined by Eq. (7)
  ρ_i, δ_i are calculated according to Eq. (6) and d_c.
  Based on ρ_i, δ_i and Eq. (8,9) to get k
  According to Eq. (10) to get L_c.
  Self-optimizing via Eq. (13) yields L_s.
  After obtaining the new loss L according to Eq. (14), the whole model is jointly optimized.

Return: Clustering results, cluster number k and hidden embedding Z.

5. Theoretical Derivations and Proofs

To maintain the coherence of the overall method presentation, we have included some theoretical content in this section. Our approach comprises two critical components: the design and underlying principles of the new loss function and the effectiveness of the new decoder. While experimental results provide evidence of their validity, theoretical derivation and formulaic proof offer stronger support for the entire methodology. Consequently, we begin by detailing the design and principles of the new loss function, followed by a comprehensive validation of the second decoder, and conclude with an analysis of its computational complexity.

5.1. Design of New Loss

We derive the loss function for the decoder. Given that reconstruction losses are computed across multiple targets, it is crucial to extract as much valuable information as possible from these targets. We have introduced a new loss function based on KL divergence [6] and InfoNCE to calculate the loss. The loss associated with the topology reconstruction of the graph is calculated using cross-entropy for the link prediction task and is denoted as L_{p_1}. Meanwhile, the loss for the node information reconstruction is computed using InfoNCE and recorded as L_{p_2}. A similar approach is discussed in Hou et al. [39], where the graph autoencoder utilizes link prediction to achieve high accuracy in reconstruction; however, it exhibits limitations in other downstream tasks, such as node classification. Therefore, we designed two distinct reconstruction tasks to derive the respective losses.

Link prediction partial loss L_{p_1}. Before delving into the design details and validity proof of the loss function, which comprises mutual information [41] and KL divergence, it is essential to establish a foundation for analyzing topological information, grounded in the homophily assumption that neighboring nodes exhibit similar semantics. By linking the edge reconstruction objective of GAE to the optimization objective of contrastive learning, an approximation of KL divergence was introduced in MINE [45] to derive an estimate for the lower bound of certainty based on mutual information.

$$\begin{aligned} I(X;Y) &= \sup_{c:X\times Y\longrightarrow R} I_c(X;Y) \\ I_c(X;Y) &= \mathbb{E}_{x,y\sim P_{XY}} c(x,y) - log\mathbb{E}_{x,y\sim P_X\times P_Y}(e^{c(x,y)}) \end{aligned} \tag{15}$$

Where $I_c(x,y)$ is referred to as the critic function, which can be represented by a neural network. Let h denote the decoder of the GAE, we can then compare the reconstruction loss of GAE for an edge (u,v) with the mutual information representation between the k-hop subgraphs (U,V) centered on these nodes, as illustrated in Eq.(18). Denote the corresponding joint and marginal distributions as P_U, P_V, and P_{UV}, respectively. It follows that the formula in parentheses in Eq.(18) serves as an empirical estimate of Eq.(16).

$$\begin{aligned} I_h^+(U;V) &= \mathbb{E}_{u,v\sim P_{UV}} logh(u,v) \\ I_h^-(U;V) &= \mathbb{E}_{u'\sim P_{U,v'\sim P_V}} log(1-h(u,v)) \\ I_h^{GAEs}(U;V) &= -(I_h^+(U;V) + I_h^-(U;V)) \end{aligned} \tag{16}$$

In other words, assuming that h has adequate representational capacity, the empirical minimum corresponding to $I_h^{GAEs}(U;V)$ can probabilistically converge to the maximum mutual information. Consequently, the process of learning h is essentially asymptotically equivalent to maximizing the mutual information between subgraphs centered on neighboring nodes. Thus, Eq.(18) can be employed to calculate the link side refactoring target losses. This is similar to GAEs using the classic encoder-decoder framework.

$$\begin{aligned} L^+ &= \frac{1}{|\Sigma^+|}\sum_{(u,v)\in\Sigma^+} \log h_\omega(z_u,z_v), \\ L^- &= \frac{1}{|\Sigma^-|}\sum_{(u',v')\in\Sigma^-} \log(1-h_\omega(z_{u'},z_{v'})), \end{aligned} \tag{17}$$

$$\begin{aligned} L_{\mathrm{p}_1} &= -\left(L^+ + L^-\right) \\ &= -\Big(\frac{1}{|\Sigma^+|}\sum_{(u,v)\in\Sigma^+} logh_w(z_u,z_v) \\ &\quad + \frac{1}{|\Sigma^-|}\sum_{(u',v')\in\Sigma^-} \log(1-h_\omega(z_{u'},z_{v'}))\Big) \end{aligned} \tag{18}$$

Here, z represents the node representation from the encoder, while Σ^+ and Σ^- denote the actual and nonexistent edges in the link prediction, respectively. In general, $\Sigma^+ = \Sigma^-$. We denote h_w as the decoder with parameters w,

which consists of an MLP and a Sigmoid layer, expressed as $h_w(z_u, z_v) = \text{Sigmoid}(\text{MLP}(z_u \circ z_v))$.

Node reconstruction partial loss L_{p_2}. For the learning of node information, we consider a scenario in which there is a single reconstruction goal, denoted as t. The objective is to optimize the encoding with respect to t so that the encoded data closely aligns with the actual data. To achieve this, our learning objective is defined as maximizing the mutual information between the current coded data S and T. This objective can be effectively optimized by maximizing the InfoNCE loss [46], and the precise computation is detailed in L_p in the main text. Here, we present the general form for clarity: $\hat{L}_{single} = InfoNCE(S, T)$.

We expand it as follows [47]. ξ is the temperature hyperparameter. P_n represents the mapping for different S.

$$L_{p_2} = exp(\frac{1}{\xi} \cdot \frac{D_n(S) \cdot P_n}{||D_n(S)|| \cdot ||P_n||}) \tag{19}$$

Similarly, we extend a reconstruction goal to multiple goals, and we derive the following general expression.

$$\hat{L}_{multi} = \sum_{i \in 2^N - 1} \hat{\lambda}_i InfoNCE(S, T_i') \tag{20}$$

Where $T = \{T_1, T_2, ..., T_n\}$ is the set of target embeddings, and the T'=$\{T_1, T_2, T_3, ..., \{T_1, T_2\}, \{T_1, T_3\}, ..., \{T_1, T_2, ..., T_n\}\}$ is the collective set formed by every subset of T.

In summary, we provide a proof and a method for calculating the reconstruction graph loss L_{p}. It is worth mentioning that we only use the single target case of L_{p_2}

5.2. Proof of Validity of EDD Decoder

Additionally, we demonstrate the effectiveness of EDD as a second decoder. Our objective is to show that L_c will eventually converge, meaning that an optimal solution can be obtained. A sufficient condition for proving the convergence of L_c is that the function is both monotonic and bounded.

The loss function L_c is shown in Eq. (10). For a given data set X, we need to divide it by the EDD algorithm into k disjoint sets $\{C_1, C_2, ..., C_k\}$, where the center of set C_k is μ_k. Suppose there is a sample point x_i that belongs to C_k. When the cluster to which it belongs remains unchanged, the loss function does not increase. However, when the cluster changes, i.e., if other high-density points $\mu_{k'}$ are closer to x_i in Euclidean distance, we can express this with the following equation.

$$||x_i - \mu_k||_2 > ||x_i - \mu_{k'}||_2 \tag{21}$$

The change of L_c at this time is

$$L_c' = L_c - ||x_i - \mu_k||_2^2 + ||x_i - \mu_{k'}||_2^2 \tag{22}$$

Obviously we get $L_c' < L_c$. So the loss function L_c decreases monotonically in this case. In addition, consider the loss function for a cluster in the general case.

$$L_{c_i} = \sum_{x \in C_i} ||x - \mu_i||_2^2 \tag{23}$$

Let $\frac{\partial L_{c_i}}{\partial \mu_i} = 0$, the calculation yields the following equation.

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x = \mu_i' \tag{24}$$

The updated mean vector is positioned at the minimum point of the function, ensuring that this step in the algorithm does not result in an increase in the loss function.

In summary, the loss function of the algorithm is monotonically decreasing and has a lower bound of 0. According to the convergence theorem of monotone bounded series, L_c converges; that is, $\lim_{N \to \infty} L_c$ exists. This implies that as the number of iterations increases, the function must converge. Therefore, we demonstrate the effectiveness of our second decoder by proving the convergence of the loss function.

5.3. Time Complexity Analysis

We focus on exploring the computational complexity of the EDD part in this section. Firstly the time complexity of the EDD algorithm to compute the distance matrix is $O(n^2)$. Secondly the time complexity of calculating the local density of nodes is $O(n)$, the time complexity of calculating the shortest distance of nodes is $O(n^2)$. The time complexity of sorting $\hat{\gamma}$ in ascending order is $O(nlogn)$, and after that the time complexity of selecting the center of the community is $O(s)$, which s is a constant term and much smaller than n. The time complexity of assigning the remaining nodes is $O(n)$. Thus, the total time complexity of the algorithm is $O(n^2 + n + n^2 + nlogn + s + n)$. Combining these analytical results, the time complexity of the EDD algorithm is found to be $O(n^2)$.

Meanwhile, we compared the results of some nonparametric traditional clustering algorithms horizontally. The time complexity of AP [48] algorithm and DBSCAN [49] are $O(n^2)$ and $O(n)$, respectively. The complexity of the OCDDP [50] algorithm is $O(nm + n^2 logn)$, where n is the number of nodes and m is the number of edges. And the complexity of the SLPA [51] algorithm is $O(n)$.

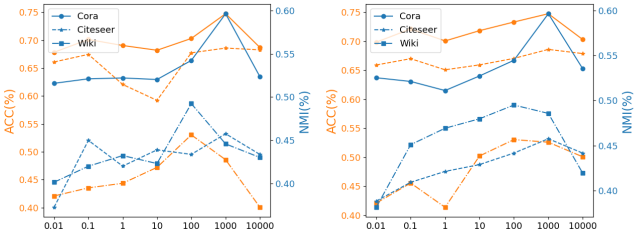
6. Experiments

6.1. Dataset and Baseline

In our experiments, we evaluated the proposed algorithm on six popular public datasets, including three graph datasets (Cora, Citeseer [52], and Wiki [53]), two non-graph datasets (USPS [54], HHAR [55]) and a larger dataset (Ogbn-arxiv [56]). The details of these datasets are summarized in Table 2. For datasets with missing adjacency matrices, we followed [20] and constructed the matrices using the heat kernel method.

Table 2
Information of dataset

Dataset	Nodes	Features	Clusters	Edges
Cora	2708	1433	7	5429
Cite	3312	3703	6	4732
Wiki	2405	4937	17	17981
USPS	9298	256	10	*
HHAR	10299	561	6	*
Ogbn-arxiv	169343	128	40	1166243



(a) α parameter sensitivity (b) β parameter sensitivity

Figure 4: The variation of index for different values of α and β

We compared our method with a variety of algorithms. Firstly, there are four nonparametric methods, namely MTL [57], DNB [15], DED [16], and DeepDPM [2]. It is important to note that these methods are not specifically designed for graph data clustering, so we made necessary modifications during runtime. Additionally, we compare DeepEDD with 10 parametric clustering methods, including TADW [58], ARGE [59], ARVGE [59], AGC [60], DAEGC [18], EGAE [19], RGC [34], FT-VGAE [4], SDCN [20], DFCN [21] and DCRN [61].

Here we adopt three public metrics to evaluate clustering performance for all compared methods, including Clustering Accuracy (ACC), Average Rand Index (ARI), Normalized Mutual Information (NMI).

6.2. Parameter Settings

In experiment, α, β are important balancing parameters. The search range of α is $\{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$. The final selection is shown in Fig.4. The selection method for β is the same. Different datasets correspond to different parameter choices, and we choose the α and β parameters when the results reach the best. Depending on the training data, the maximum number of epochs for training is set to 100 to 1000. The pre-training method is to first train the mask network and AE network for 15 rounds, and then jointly train the model for 30 rounds. The learning rate for fine-tuning is set to 10^{-2}. The dimension of the embedding layer is set to 64. All training is completed on the PyTorch platform with the NVIDIA GeForce RTX 3080.

6.3. Results and Analysis

We first compared the accuracy of predicting k using nonparametric methods, as illustrated in Table 3. We conducted 10 runs on the dataset, presenting both the accuracy and the average prediction for k. The results indicate that the performance and prediction accuracy of our method significantly outperform those of other baselines. This improvement is attributable to our method's comprehensive consideration of both topological and node information, integrated into the neural network and the learning component of EDD.

The experimental results for clustering performance are presented in Table 4, leading to the following conclusions: 1) DeepEDD outperforms methods based on separate GNNs on the dataset, primarily because our approach incorporates both attribute and structural information for clustering. For example, with respect to the NMI metric, its performance increases exceed that of EGAE by 6.24%, 4.68%, 1.54%, and 1.80%, respectively. 2) Furthermore, regarding the suboptimal performance of ACC on certain datasets, we attribute this to the higher ratio of the number of nodes to the dimensions of node features in these datasets. This suggests that attribute information plays a more critical role in clustering performance than structural information. Some baselines integrate additional architectures to extract attribute information, resulting in better-learned attribute embeddings, as observed in methods such as DFCN and SDCN. However, our advantage lies in being a nonparametric method, which eliminates the need to input the value of k in advance, as required in parametric methods.

We subsequently explored the significance of the k value to emphasize the superiority of DeepEDD's nonparametric approach. The relevant results are presented in Table 5. We investigate the impact of deviations in the value of k at 1, 2, and 3 on the Cora, Citeseer, and Wiki datasets, respectively. The results indicate that deviations in the k value have a significant effect on parameter-based algorithms; however, this adverse effect does not occur with the DeepEDD algorithm, which achieves the best results. This advantage arises because the EDD algorithm itself does not rely on the hyperparameter k. The enforced setting of k only affects the neural network component of the model.

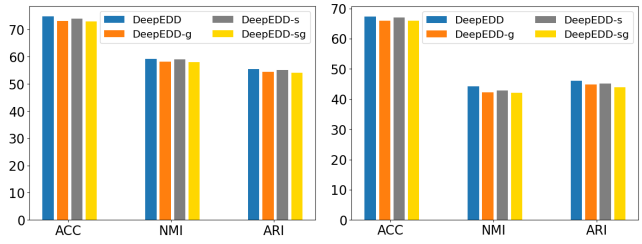
To further investigate the robustness and generalization of our algorithm, we conducted experiments on non-graph datasets, as presented in Table 6. For the dataset lacking an affinity matrix, we followed Bo et al. [20] to construct the matrix using the heat kernel method. The results indicate that, as a nonparametric algorithm specifically designed for graph-structured data, our method can effectively predict the cluster number k. The accuracy achieved is at least 10% higher than that of the runner-up best method. Furthermore, we applied the entire framework to other tasks, such as node classification and link prediction. The average performance ranking is first place. These results unequivocally demonstrate the efficacy of the mask as an auxiliary task for self-supervised learning. Not only does it significantly reduce

Table 3
Average k prediction accuracy on 6 benchmarks. Bold and underlined values indicate the best and runner-up results, respectively. Mean represents the average of k's prediction. Accuracy represents the accuracy of k's correct prediction.

Methods	MtL		DNB		DED		DeepDPM		RGC		DeepEDD	
	mean	Accuracy	mean	Accuracy	mean	Accuracy	mean	Accuracy	mean	Accuracy	mean	Accuracy
Cora	3.0(±1.0)	0/10	6.0(±0.0)	0/10	10.2(±0.7)	0/10	5.1(±2.3)	1/10	7.1(±0.2)	10/10	7.0(±0.3)	**10/10**
Citeseer	6.7(±3.0)	1/10	5.0(±0.0)	0/10	8.0(±2.0)	1/10	4.0(±5.2)	2/10	5.6(±0.4)	9/10	5.9(±0.6)	**10/10**
Wiki	2.1(±3.8)	0/10	2.0(±0)	0/10	2.8(±2.1)	0/10	3.0(±11.3)	0/10	15.0(±1.1)	6/10	17.3(±0.8)	**8/10**
USPS	1.8(±3.2)	1/10	2.9(±0.3)	0/10	4.3(±2.0)	1/10	6.0(±1.9)	2/10	13.0(±3.8)	5/10	10.8(±1.2)	**7/10**
HHAR	0.9(±1.8)	0/10	3.0(±0.2)	1/10	2.3(±2.9)	0/10	9.3(±7.4)	1/10	5.0(±3.2)	4/10	7.3(±1.2)	**7/10**
Ogbn-arxiv	20.4(±12.1)	0/10	20.0(±0)	0/10	32.0(±14.9)	0/10	50.0(±20.3)	0/10	32.0(±12.3)	0/10	48.2(±10.1)	**3/10**

Table 4
Clustering performance results on 6 benchmarks. Bold and underlined values indicate the best and runner-up results, respectively(%).

Dataset	Index	TADW	ARGE	ARVGE	AGC	DAEGC	SDCN	DFCN	EGAE	DCRN	RGC	FT-VGAE	DeepEDD-A	DeepEDD
Cora	ACC	56.03	64.00	63.80	68.92	70.40	71.00	74.02	72.42	73.34	71.58	**76.20**	73.82	75.74
	NMI	44.11	44.90	45.00	53.68	52.80	50.25	53.90	53.96	52.00	57.60	58.20	58.00	**60.20**
	ARI	33.20	35.20	37.40	-	49.60	47.02	48.10	47.22	50.10	49.46	56.70	56.04	**58.37**
Citeseer	ACC	45.48	57.30	54.40	67.00	67.20	66.00	69.50	67.42	69.51	67.22	**69.53**	67.20	67.30
	NMI	29.14	35.00	26.10	41.13	39.70	38.70	43.90	41.18	45.80	45.06	45.20	45.00	**45.86**
	ARI	22.81	34.10	24.50	-	41.00	40.20	45.50	43.18	46.64	46.17	45.60	45.05	**46.70**
Wiki	ACC	31.01	38.01	38.34	47.65	49.10	51.21	51.08	51.19	51.70	50.02	52.40	49.90	**53.02**
	NMI	25.07	34.17	32.81	45.25	28.05	40.13	39.85	47.47	42.31	43.00	45.02	48.00	**49.01**
	ARI	4.23	10.09	10.63	-	32.80	30.91	33.07	33.08	33.27	30.10	34.21	34.18	**35.00**
HHAR	ACC	50.31	59.02	60.00	62.00	75.00	84.00	86.01	78.53	85.32	83.99	**86.12**	85.20	86.00
	NMI	42.08	48.20	51.03	53.06	68.50	80.00	81.38	69.22	80.30	80.02	81.00	82.02	**82.10**
	ARI	22.23	31.80	40.00	-	59.90	72.70	76.13	77.63	69.03	78.21	78.02	78.10	**78.11**
USPS	ACC	51.20	62.09	61.79	67.25	73.62	78.11	79.40	74.97	**79.70**	73.20	79.12	78.94	79.05
	NMI	42.28	48.11	41.02	53.02	71.10	74.02	70.30	73.22	78.04	80.03	81.63	79.82	**81.70**
	ARI	24.23	31.04	30.00	-	63.30	70.19	70.21	69.03	69.23	70.29	72.39	73.00	**73.02**
Ogbn-arxiv	ACC	1.22	3.21	2.02	12.84	29.40	30.10	**31.00**	30.11	29.70	23.90	30.12	28.94	28.95
	NMI	12.18	8.26	11.00	23.06	40.00	44.09	40.00	43.20	40.00	40.03	44.60	44.82	**45.00**
	ARI	0.23	1.84	10.05	-	19.30	20.19	20.01	20.00	19.15	20.20	12.42	20.10	**22.03**



(a) Cora (b) Citeseer

Figure 5: Explore the effect of different settings on the overall performance of DeepEDD. DeepEDD-sg represents the absence of both modules, the self-optimization, and GAT(%).

the need for manual annotation, but it also substantially enhances the algorithm's generalization capabilities.

6.4. Ablation Study

In the preceding subsection, we briefly examined the impact of fused AE coding on the overall algorithm, specifically the outcomes of DeepEDD-A. We reference previous work [62, 63, 64] regarding the setup of ablation experiments and conduct an ablation study to validate the effectiveness of each component in our method. This analysis encompasses the GAT layer with a masking strategy, the self-optimization module, the selection of reconstruction targets, and the evaluation of generalization performance, among others.

As depicted in Fig.5, DeepEDD-s represents the version of DeepEDD that omits the self-optimization step, DeepEDD-g represents the version that replaces the GAT layer with GCN, and DeepEDD-sg represents the version that replaces both components simultaneously. The performance of DeepEDD-s is inferior across both datasets. The results for DeepEDD-g show that while the decrease in ACC values is not significant, both NMI and ARI values exhibit a noticeable decline, indicating a reduction in interpretability and generalization performance.

Subsequently, we conducted a comprehensive analysis of the reconstructed targets through dedicated experiments, as outlined in Table 7. We identified distinct reconstruction targets using DeepEDD-N, DeepEDD-E, and DeepEDD-G to represent the Node2vec-only target, AE structure-only target, and GAE overall structure-only target, respectively. Additionally, DeepEDD-All denotes the reconstruction aimed at amalgamating these targets. Upon analyzing the data in the table, the following conclusions can be drawn: the performance of the three models, DeepEDD-N, DeepEDD-E, and DeepEDD-G, is comparable, while both DeepEDD and DeepEDD-All demonstrate improvements of

Table 5
The effect of different k value choices on different baseline. Bold and underlined values indicate the best and runner-up results, respectively(%).

Dataset	k	Index	TADW	ARGE	ARVGE	AGC	DAEGC	SDCN	DFCN	EGAE	DCRN	RGC	FT-VGAE	DeepEDD-A	DeepEDD
Cora	k=6	ACC	48.13	55.00	55.80	58.33	61.30	61.60	62.01	58.83	63.11	64.01	62.12	66.12	**68.21**
		NMI	34.01	35.10	36.01	31.61	32.81	30.36	23.10	33.57	42.01	50.85	43.31	50.00	**51.30**
		ARI	23.11	25.24	27.41	-	29.15	26.12	22.30	27.15	30.30	40.09	31.50	41.01	**42.09**
	k=8	ACC	45.10	53.60	54.10	56.00	60.00	61.74	64.70	57.81	62.01	63.92	62.00	65.10	**68.02**
		NMI	32.00	33.40	35.00	33.13	32.01	30.30	21.00	33.80	41.07	50.63	42.20	49.03	**51.63**
		ARI	21.01	22.29	21.51	-	23.71	27.17	23.10	24.10	31.00	39.06	31.00	41.00	**43.00**
Citeseer	k=4	ACC	28.10	34.16	35.60	41.90	45.27	46.70	49.10	52.00	51.73	56.10	52.90	63.00	**64.10**
		NMI	14.12	23.70	11.13	25.10	24.73	19.32	28.04	27.14	29.17	39.04	30.00	39.94	**41.06**
		ARI	4.31	12.15	10.40	-	20.87	20.50	21.20	23.00	26.23	33.08	28.15	37.05	**37.76**
	k=8	ACC	26.90	36.73	33.64	40.00	45.58	46.86	46.80	52.80	52.65	53.10	54.50	62.04	**63.85**
		NMI	15.10	23.90	10.11	22.14	20.70	15.88	25.02	27.85	29.60	38.06	30.40	37.90	**40.85**
		ARI	3.00	9.97	10.47	-	18.55	19.57	20.23	22.80	26.70	32.03	27.95	36.00	**36.03**
Wiki	k=14	ACC	3.00	7.41	10.23	19.03	20.00	21.62	21.56	24.09	25.10	20.01	22.69	36.01	**37.08**
		NMI	2.03	5.07	4.85	14.65	4.90	13.10	12.09	15.37	13.00	26.04	16.00	30.70	**32.68**
		ARI	1.53	3.32	4.71	-	6.89	4.93	9.04	16.34	13.20	11.90	14.60	23.10	**25.00**
	k=20	ACC	2.05	5.70	7.10	13.51	20.30	20.02	20.50	21.63	24.15	19.91	21.00	34.06	**36.98**
		NMI	2.00	6.00	5.15	15.03	4.21	12.00	10.03	14.30	13.79	29.10	14.20	29.90	**31.42**
		ARI	1.01	4.23	4.00	-	6.90	3.95	9.00	15.61	14.26	20.80	14.65	24.04	**25.10**

Table 6
Generalization and robustness testing results. Mean represents the average of k's prediction. Accuracy represents the accuracy of k's correct prediction.

Methods	USPS		HHAR	
	Mean	Accuracy	Mean	Accuracy
MtL	7.0(±2.0)	5/10	7.2(±1.0)	3/10
DNB	8.0(±4.0)	4/10	5.0(±0.0)	0/10
DED	11.0(±1.1)	4/10	9.0(±2.2)	6/10
RGC	9.5(±0.5)	9/10	5.8(±0.6)	8/10
DeepDPM	9.4(±1.0)	9/10	6.0(±0.9)	9/10
DeepEDD	10.0(±0.0)	10/10	6.0(±0.3)	9/10

Table 7
Performance for different reconstruction target models(%).

Methods	Cora			Citeseer		
	ACC	NMI	ARI	ACC	NMI	ARI
DeepEDD-N	73.88	58.68	54.90	67.00	43.91	45.82
DeepEDD-E	73.15	58.60	55.00	65.06	43.86	45.57
DeepEDD-G	74.70	59.06	57.10	66.05	44.90	45.93
DeepEDD-All	75.75	60.10	58.30	67.33	45.32	46.04
DeepEDD	75.74	60.20	58.37	67.30	45.86	46.70

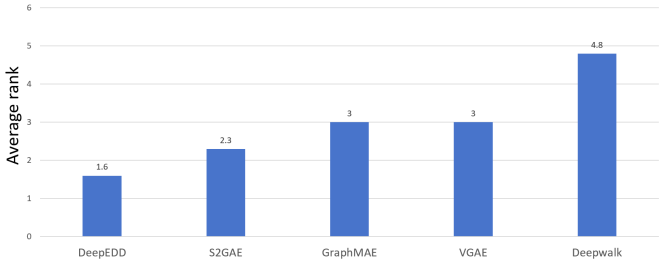


Figure 6: The average rank of the algorithm for the link prediction and node classification tasks.

approximately 1% across all three indicators, making them comparable. However, DeepEDD-All aims to rebuild GAE, which leads to redundant node and edge reconstruction, resulting in high computational complexity and wasted computing power; thus, DeepEDD-All is not utilized as the default setting.

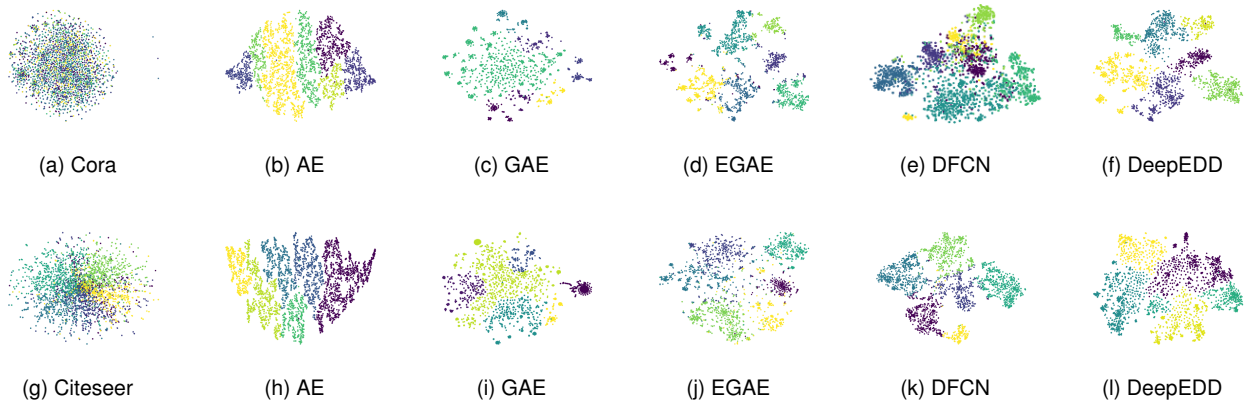
We conducted a comprehensive exploration of various experimental default settings. The experiments included selecting the number of encoder layers, the sub-distribution selection for the self-optimizing partial Q-distribution, and evaluating the performance of the enhanced and optimized density clustering algorithm. Additionally, we incorporated node classification and link prediction tasks within the DeepEDD framework to demonstrate the improved generalization ability of our model. Our framework consistently achieves the best or best-equivalent results across all tasks.

The data on the generalization performance experiment is shown in Fig.6. The specific experimental design links prediction and node classification, and the results are presented in Tables 8 and 9.

Finally, we present the following supplementary experiments to further investigate the optimal experimental parameters and clustering visualization. Tables 10, 11, and 12 display the results of the ablation experiments conducted to explore the parameters of the experimental setup. Overall, we achieved the best results using a default Q distribution with two GAT layers, in conjunction with our method for automatically determining d_c. We also show the impact of CH values in Table 13. It can be seen that proper invocation of inter-class information in CH is conducive to obtaining more accurate k value. All experiments were conducted on the Cora dataset.

6.5. t-SNE Visualization

In Fig.7, we employ t-SNE [44] to visualize the original distributions of the Cora and Citeseer datasets, as well as to compare the post-clustering distributions of AE, GAE, EGAE, DFCN, and DeepEDD. The results clearly indicate



(a) Cora (b) AE (c) GAE (d) EGAE (e) DFCN (f) DeepEDD

(g) Citeseer (h) AE (i) GAE (j) EGAE (k) DFCN (l) DeepEDD

Figure 7: 2D visualization on Cora and Citeseer

Table 8
Experimental results (ACC%) for the node classification task.

Datasets	Cora	Citeseer	DBLP
Deepwalk	73.32	48.24	76.37
VGAE	80.60	71.85	78.21
GraphMAE	84.22	**72.74**	80.57
S2GAE	84.14	72.21	80.14
DeepEDD	**84.60**	71.10	**81.10**

Table 9
Link prediction task experiment results (ACC%) where the default metrics are AP.

Datasets	Cora	Citeseer	DBLP
Deepwalk	81.77	72.77	91.76
VGAE	95.12	93.80	97.75
GraphMAE	88.32	91.54	68.60
S2GAE	93.96	94.22	**98.76**
DeepEDD	**95.15**	**94.32**	93.31

that DeepEDD effectively reveals the inherent clustering structure among the samples, resulting in looser inter-cluster distances and more compact intra-cluster formations.

6.6. Limitation

Experimental results show that while DeepEDD achieves optimal performance on benchmarks such as Ogbn-arxiv, it exhibits certain limitations when applied to very large-scale graphs. Specifically, the algorithm's inherent computational complexity of $O(n^2)$ makes it less suitable for graphs with hundreds of thousands of nodes, leading to significant computational and memory bottlenecks. To address this, targeted strategies must be adopted to enhance the scalability and efficiency of DeepEDD. Potential solutions include: 1) retaining only the top- k neighbors for each node during message passing to reduce unnecessary computations; 2) leveraging graph partitioning techniques to divide the original graph into manageable subgraphs, computing local embeddings independently, and aggregating them via a hierarchical module to reduce global overhead; and 3) designing dynamic batch processing strategies that exploit graph sparsity by focusing computation on node pairs with high local correlation. These strategies not only offer directions for improving DeepEDD's efficiency but also highlight promising avenues for future research—such as the development of a dynamic graph-processing module or the integration of a linear-complexity core to further reduce the algorithm's overall computational burden. In addition, exploring the effective integration of DeepEDD into real-world applications represents an important direction for future research. Potential use cases include user group clustering in recommendation systems [65] and the construction of user profiles in e-commerce environments [66], where the ability to capture complex structural information and automatically determine clusters could provide significant practical value.

7. Conclusion

In this paper, we propose a new nonparametric deep graph clustering method, DeepEDD, whose core components are a fusion graph masking autoencoder and a novel algorithm based on effective distance and density, referred to as EDD. In this context, we distinguish our approach from the original method through the incorporation of innovative mechanisms, notably cluster number estimators and determiners. Mathematical calculations and experiments demonstrate that DeepEDD outperforms existing parametric and nonparametric clustering methods. We also illustrate the added value that nonparametric methods contribute to deep clustering in terms of sensitivity to and importance of the hypothesis k. Furthermore, we have demonstrated the generalization capabilities of DeepEDD, which is the first of its kind to perform connection prediction and node classification tasks. However, because of the mask mechanism, the existing data reconstruction process depends on the quality of the target data. In addition, the hyperparameters in the process of coding fusion also need to be well set. Therefore,

Table 10
Q_g stands for using only Z_m as the Q-distribution.The Q_{ag} representation uses Z_{ae} and Z_m. Q stands for the default setting(%).

Methods	ACC	NMI	ARI
Q_g	74.64	56.60	55.00
Q_{ag}	73.28	55.30	55.64
Q	75.74	60.20	58.37

Table 11
We conducted experiments using 1 Layer,2 Layers (default settings), and 4 Layers GAT as the basis of the mask encoder(%).

Methods	ACC	NMI	ARI
1 layer	67.00	49.80	53.41
2 layers	75.74	60.20	58.37
4 layers	75.05	56.00	58.01

Table 12
The effect of different truncation distances d. d_I stands for before improvement. Experimentation in 10 and 20 rounds. Mean represents the average of k's prediction. Accuracy represents the accuracy of k's correct prediction.

Methods	Mean	K
d_c	7.80	7/10
d_I	7.10	10/10
d_I	7.17	19/20

Table 13
The influence of CH value on prediction accuracy. Mean represents the average of k's prediction. Accuracy represents the accuracy of k's correct prediction.

Methods	Mean	K
$CH(\nu = 0.9)$	8.02	7/10
$CH(\nu = 0.8)$	8.70	5/10
Without CH	6.92	9/10

the future work will focus on how to achieve the non-parameterization of each module.

CRediT authorship contribution statement

Yuanchi Ma: Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Hui He**: Validation, Software, Investigation. **Zhongxiang Lei**: Conceptualization. **Kaize Shi**: Writing – review & editing. **Xueping Peng**: Writing – review & editing. **Zhendong Niu**: Writing – review & editing, Validation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62272048 and the National Key Research and Development Program of China under Grant 2019YFB1406302.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.
[2] M. Ronen, S. E. Finder, O. Freifeld, Deepdpm: Deep clustering with an unknown number of clusters, in: CVPR, IEEE, 2022, pp. 9851–9860.
[3] J. Chang, J. W. F. III, Parallel sampling of DP mixture models using sub-cluster splits, in: NIPS, 2013, pp. 620–628.
[4] N. Mrabah, M. Bouguessa, R. Ksantini, Escaping feature twist: A variational graph auto-encoder for node clustering, in: L. D. Raedt (Ed.), Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, International Joint Conferences on Artificial Intelligence Organization, 2022, pp. 3351–3357. Main Track.
[5] J. Xie, R. B. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: ICML, volume 48 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2016, pp. 478–487.
[6] T. N. Kipf, M. Welling, Variational graph auto-encoders, CoRR abs/1611.07308 (2016).
[7] A. Hasanzadeh, E. Hajiramezanali, K. R. Narayanan, N. Duffield, M. Zhou, X. Qian, Semi-implicit graph variational auto-encoders, in: NeurIPS, 2019, pp. 10711–10722.
[8] T. Zhao, Z. Wang, A. Masoomi, J. G. Dy, Adaptive nonparametric variational autoencoder, CoRR abs/1906.03288 (2019).
[9] S. Lukasik, P. A. Kowalski, M. Charytanowicz, P. Kulczycki, Clustering using flower pollination algorithm and calinski-harabasz index, in: CEC, IEEE, 2016, pp. 2724–2728.
[10] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.
[11] A. Rodriguez, A. Laio, "clustering by fast search and find of density peaks., Science 344 (2014) 1492.
[12] X. Wei, M. Peng, H. Huang, Y. Zhou, An overview on density peaks clustering, Neurocomputing 554 (2023) 126633.
[13] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, CoRR abs/2006.04131 (2020).
[14] S. A. Shah, V. Koltun, Deep continuous clustering, CoRR abs/1803.01449 (2018).
[15] Z. Wang, Y. Ni, B. Jing, D. Wang, H. Zhang, E. P. Xing, DNB: A joint learning framework for deep bayesian nonparametric clustering, IEEE Trans. Neural Networks Learn. Syst. 33 (2022) 7610–7620.
[16] Y. Wang, Z. Shi, X. Guo, X. Liu, E. Zhu, J. Yin, Deep embedding for determining the number of clusters, in: AAAI, AAAI Press, 2018, pp. 8173–8174.
[17] R. A. Walker, S. Shah, N. Gupta, Computer-aided engineering (cae) for system analysis, Proceedings of the IEEE 72 (1984) 1732–1745.
[18] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, C. Zhang, Attributed graph clustering: A deep attentional embedding approach, in: IJCAI, ijcai.org, 2019, pp. 3670–3676.

[19] H. Z. R. Zhang, X. Li, Embedding graph auto-encoder for graph clustering, in: IEEE Transactions on Neural Networks and Learning Systems, volume 45, 2023, pp. 3986–3993.

[20] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: WWW, ACM / IW3C2, 2020, pp. 1400–1410.

[21] W. Tu, S. Zhou, X. Liu, X. Guo, Z. Cai, E. Zhu, J. Cheng, Deep fusion clustering network, in: AAAI, AAAI Press, 2021, pp. 9978–9987.

[22] K. Hassani, A. H. K. Ahmadi, Contrastive multi-view representation learning on graphs, in: ICML, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 4116–4126.

[23] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, in: ICLR (Poster), OpenReview.net, 2019.

[24] N. Park, R. A. Rossi, E. Koh, I. A. Burhanuddin, S. Kim, F. Du, N. K. Ahmed, C. Faloutsos, CGC: contrastive graph clustering for community detection and tracking, CoRR abs/2204.08504 (2022).

[25] X. Yang, Y. Liu, S. Zhou, S. Wang, W. Tu, Q. Zheng, X. Liu, L. Fang, E. Zhu, Cluster-guided contrastive graph clustering network, in: AAAI, AAAI Press, 2023, pp. 10834–10842.

[26] Y. Zheng, C. Jia, J. Yu, Attributed graph clustering under the contrastive mechanism with cluster-preserving augmentation, Inf. Sci. 681 (2024) 121225.

[27] T. Wang, J. Wu, Y. Qi, X. Qi, J. Guan, Y. Zhang, G. Yang, Neighborhood contrastive representation learning for attributed graph clustering, Neurocomputing 562 (2023) 126880.

[28] Y. Liu, X. Yang, S. Zhou, X. Liu, S. Wang, K. Liang, W. Tu, L. Li, Simple contrastive graph clustering, IEEE Transactions on Neural Networks and Learning Systems (2023) 1–12.

[29] E. Pan, Z. Kang, Multi-view contrastive graph clustering, in: NeurIPS, 2021, pp. 2148–2159.

[30] O. Maddouri, X. Qian, B.-J. Yoon, Geometric affinity propagation for clustering with network knowledge, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 11419–11436.

[31] G. Xu, S. Huang, A novel non-maximum suppression algorithm based on affinity propagation clustering, in: 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), 2021, pp. 270–276. doi:10.1109/ICPS49255.2021.9468162.

[32] K. Berahmand, M. Mohammadi, R. Sheikhpour, Y. Li, Y. Xu, WSNMF: weighted symmetric nonnegative matrix factorization for attributed graph clustering, Neurocomputing 566 (2024) 127041.

[33] B. Li, Z. Shu, Y. Liu, C. Mao, S. Gao, Z. Yu, Multi-view clustering via label-embedded regularized NMF with dual-graph constraints, Neurocomputing 551 (2023) 126521.

[34] Y. Liu, K. Liang, J. Xia, X. Yang, S. Zhou, M. Liu, X. Liu, S. Z. Li, Reinforcement graph clustering with unknown cluster number, in: ACM Multimedia, ACM, 2023, pp. 3528–3537.

[35] S. M. Ko, S. Cho, D.-W. Jeong, S. Han, M. Lee, H. Lee, Grouping matrix based graph pooling with adaptive number of clusters 37 (2023) 8334–8342.

[36] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT (1), Association for Computational Linguistics, 2019, pp. 4171–4186.

[37] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, R. B. Girshick, Masked autoencoders are scalable vision learners, in: CVPR, IEEE, 2022, pp. 15979–15988.

[38] J. Li, R. Wu, W. Sun, L. Chen, S. Tian, L. Zhu, C. Meng, Z. Zheng, W. Wang, What's behind the mask: Understanding masked graph modeling for graph autoencoders, in: KDD, ACM, 2023, pp. 1268–1279.

[39] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, J. Tang, Graphmae: Self-supervised masked graph autoencoders, in: KDD, ACM, 2022, pp. 594–604.

[40] Q. Tan, N. Liu, X. Huang, S. Choi, L. Li, R. Chen, X. Hu, S2GAE: self-supervised graph autoencoders are generalizable learners with graph masking, in: WSDM, ACM, 2023, pp. 787–795.

[41] K. Hassani, A. H. K. Ahmadi, Contrastive multi-view representation learning on graphs, in: ICML, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 4116–4126.

[42] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: KDD, ACM, 2016, pp. 855–864.

[43] Y. Shi, Y. Dong, Q. Tan, J. Li, N. Liu, Gigamae: Generalizable graph masked autoencoder via collaborative latent space reconstruction, CoRR abs/2308.09663 (2023).

[44] L. van der Maaten, G. offrey Hinton., Visualizing data using t-sne, in: JMLR, 2008, p. 2579–2605.

[45] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, R. D. Hjelm, A. C. Courville, Mutual information neural estimation, in: ICML, volume 80 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 530–539.

[46] A. van den Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, CoRR abs/1807.03748 (2018).

[47] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, in: ICLR (Poster), OpenReview.net, 2019.

[48] B. J. FREY, D. DUECK, Clustering by passing messages between data points, in: Science, volume 315, 2007, pp. 972–976.

[49] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, AAAI Press, 1996, pp. 226–231.

[50] X. Bai, P. Yang, X. Shi, An overlapping community detection algorithm based on density peaks, Neurocomputing 226 (2017) 7–15.

[51] J. Xie, B. K. Szymanski, Towards linear time overlapping community detection in social networks, in: Proc. 16th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining, 2012, p. 25–36.

[52] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (2008) 93–106.

[53] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Y. Chang, Network representation learning with rich text information, in: IJCAI, AAAI Press, 2015, pp. 2111–2117.

[54] J. J. Hull, A database for handwritten text recognition research, IEEE Transactions on pattern analysis and machine intelligence 16 (1994) 550–554.

[55] C. Li, D. Niu, B. Jiang, X. Zuo, J. Yang, Meta-har: Federated representation learning for human activity recognition, in: Proc. World Wide Web Conf, 2021, pp. 912–922.

[56] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, in: NeurIPS, 2020.

[57] B. A. Pimentel, A. C. de Carvalho, A meta-learning approach for recommending the number of clusters for clustering algorithms, Knowledge-Based Systems 195 (2020) 105682.

[58] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Y. Chang, Network representation learning with rich text information, in: IJCAI, AAAI Press, 2015, pp. 2111–2117.

[59] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: IJCAI, ijcai.org, 2018, pp. 2609–2615.

[60] X. Zhang, H. Liu, Q. Li, X. Wu, Attributed graph clustering via adaptive graph convolution, in: IJCAI, ijcai.org, 2019, pp. 4327–4333.

[61] Y. Liu, W. Tu, S. Zhou, X. Liu, L. Song, X. Yang, E. Zhu, Deep graph clustering via dual correlation reduction, in: AAAI, AAAI Press, 2022, pp. 7603–7611.

[62] A. A. Baffour, Z. Qin, J. Geng, Y. Ding, F. Deng, Z. Qin, Generic network for domain adaptation based on self-supervised learning and deep clustering, Neurocomputing 476 (2022) 126–136.

[63] H. He, Q. Zhang, S. Bai, K. Yi, Z. Niu, CATN: cross attentive tree-aware network for multivariate time series forecasting, in: AAAI, AAAI Press, 2022, pp. 4030–4038.

[64] K. Shi, X. Sun, Q. Li, G. Xu, Compressing long context for enhancing RAG with amr-based concept distillation, CoRR abs/2405.03085 (2024).

[65] V. N. Ioannidis, A. S. Zamzam, G. B. Giannakis, N. D. Sidiropoulos, Coupled graphs and tensor factorization for recommender systems and community detection, IEEE Trans. Knowl. Data Eng. 33 (2021) 909–920.

[66] K. Shi, X. Sun, D. Wang, Y. Fu, G. Xu, Q. Li, Llama-e: Empowering e-commerce authoring with object-interleaved instruction following, in: COLING, Association for Computational Linguistics, 2025, pp. 870–885.

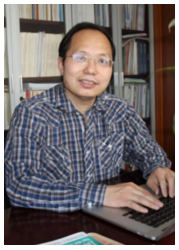
Yuanchi Ma received his Masters Degree from the University of Auckland, Auckland, NZ in 2021. He is currently pursuing his Ph.D. in the School of Computer Science at the Beijing Institute of Technology, Beijing, China. His current research interests are in cluster analysis and knowledge services.

Hui He received the M.E. degree from University of Shanghai for Science and Technology, Shanghai, China in 2020. She is currently pursuing the Ph.D. degree at Institute of Engineering Medicine, Beijing Institute of Technology, Beijing, China. Her current research interests focus on multivariate time-series analysis and knowledge services.

Kaize Shi (S'20-M'21) is with the Data Science and Machine Intelligence Lab, University of Technology Sydney. He has PhD degrees in computer science and computer systems, which are from the Beijing Institute of Technology, China, and the University of Technology Sydney, Australia. His research interests include natural language generation, social computing, cyber-physical-social systems, meteorological knowledge services, intelligent transportation, and artificial intelligence technology. He is the associate editor of IEEE Transactions on Computational Social Systems and academic editor of PeerJ Computer Science and Wireless Communications and Mobile Computing. He also served as a guest editor for the International Journal of Distributed Sensor Networks, and as a reviewer for the IEEE Transactions on Intelligent Transportation Systems, IEEE Internet of Things Journal, etc. He served as a program committee member for conferences of ACL, EMNLP, SIGKDD, ICDM, etc. He is a member of the Artificial Intelligence Technical Committee of the China Meteorological Service Association.

Zhongxiang Lei is pursuing studies at Beijing Institute of Technology since 2022. Research areas include federated learning and non-convex optimization, with a keen interest in large-scale models for natural language processing.

Xueping Peng (Member, IEEE) received the Ph.D.degree in computer science from the University of Technology Sydney (UTS), Sydney, NSW, Australia, in 2015. He is currently a Lecturer with the Australian Artificial Intelligence Institute, UTS, where he was also previously a Post-Doctoral Research Fellow. He is currently conducting application-driven research focusing on artificial intelligence, data mining, social network analysis, and intelligent healthcare.



Zhendong Niu received a Ph.D. degree in computer science from the Beijing Institute of Technology in 1995. He was a Post-Doctoral Researcher with the University of Pittsburgh from 1996 to 1998, a Researcher & Adjunct from 1999 to 2004, and a Joint Research Professor with the Information School, University of Pittsburgh since 2006. He served as the deputy dean for the School of Software from 2002 to 2006, the deputy dean for the School of Computer Science and Technology from 2006 to 2019, and the director of the Library from 2019 to 2021, Beijing Institute of Technology. His research interests include informational retrieval, software architecture, digital libraries, and web-based learning techniques. Prof.Niu is a recipient of the IBM Faculty Innovation Award in 2005 and the New Century Excellent Talents at the University of Ministry of Education of China in 2006.