CaEGCN: Cross-Attention Fusion Based Enhanced Graph Convolutional Network for Clustering

Guangyu Huo, Yong Zhang, *Member, IEEE*, Junbin Gao, Boyue Wang, Yongli Hu, and Baocai Yin

Abstract—With the powerful learning ability of deep convolutional networks, deep clustering methods can extract the most discriminative information from individual data and produce more satisfactory clustering results. However, existing deep clustering methods usually ignore the relationship between the data. Fortunately, the graph convolutional network can handle such relationships, opening a new research direction for deep clustering. In this paper, we propose a cross-attention based deep clustering framework, named Cross-Attention Fusion based Enhanced Graph Convolutional Network (CaEGCN), which contains four main modules: the cross-attention fusion module which innovatively concatenates the Content Auto-encoder module (CAE) relating to the individual data and Graph Convolutional Auto-encoder module (GAE) relating to the relationship between the data in a layer-by-layer manner, and the self-supervised model that highlights the discriminative information for clustering tasks. While the cross-attention fusion module fuses two kinds of heterogeneous representation, the CAE module supplements the content information for the GAE module, which avoids the over-smoothing problem of GCN. In the GAE module, two novel loss functions are proposed that reconstruct the content and relationship between the data, respectively. Finally, the self-supervised module constrains the distributions of the middle layer representations of CAE and GAE to be consistent. Experimental results on different types of datasets prove the superiority and robustness of the proposed CaEGCN.

Index Terms—Cross-attention fusion mechanism, graph convolutional network, deep clustering

1 Introduction

Clustering is an essential topic in the data mining area, which divides a collection of objects into multiple clusters of similar objects. Inspired by the powerful feature extraction capability of the deep convolutional network, many deep learning based clustering methods have been proposed in recent years, demonstrating much significant progress in clustering research [1], [2], [3], [4]. The two-step spectral clustering method is usually employed here: A 'good' data representation or similarity matrix which is learned from these deep learning methods can be pipelined to the downstream models/algorithms such as K-means [5] or Normalized Cut [6] to obtain the final clustering result.

- *Guangyu Huo, Yong Zhang, Boyue Wang, Yongli Hu, and Baocai Yin are with the Beijing Key Laboratory of Multimedia and Intelligent Software Technology, Beijing Artificial Intelligence Institute, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China. E-mail: gyhuo@emails.bjut.edu.cn, {zhangyong2010, wby, huyongli, ybc}@bjut.edu.cn.*
- *Junbin Gao is with Discipline of Business Analytics, The University of Sydney Business School, The University of Sydney, Sydney, NSW 2006, Australia. E-mail: junbin.gao@sydney.edu.au.*

Manuscript received 29 Dec. 2020; revised 14 July 2021; accepted 27 Oct. 2021. Date of publication 4 Nov. 2021; date of current version 7 Mar. 2023.
The work was supported in part by the National Natural Science Foundation of China under Grants 62072015, 61906011, U19B2039, U1811463, and 61632006, in part by Beijing Natural Science Foundation under Grant 4204086, and in part by Beijing Municipal Science and Technology through Project No. KM202010005014.
(Corresponding author: Boyue Wang.)
Recommended for acceptance by Q. He.
Digital Object Identifier no. 10.1109/TKDE.2021.3125020

However, existing deep clustering methods only focus on the data content, and usually ignore the relationship between the data, i.e., the structural information. With the development of the data collection and analysis technologies, people not only collect the data but also obtain or build the relationship between the data in the form of graphs, such as social networks [7], biochemical structure networks [8] and railway networks [9]. These graphs can help people make better data-driven decisions. Therefore, how to embed the relationship between the data into deep clustering becomes a thorny problem.

Furthermore, based on these raw graphs, one wishes to mine the latent relationship between the data effectively. As we know, the edges in a graph represent the explicit relationship, which is also regarded as the first-order structural relationship. Many graph embedding methods exploit such relationship, including DeepWalk [10], node2vec [11], and LINE [12]. But, the data relationship in the real-world is complicated. There still exist many implicit and complicated relationships. For example, two nodes may not be directly connected in a graph, while they have many identical neighbors. However, it is a natural belief that these two nodes have a high-order structural relationship.

In order to improve the clustering effect of deep clustering methods, utilizing the high-order relationships is necessary. As an important approach in deep learning methods, Graph Convolutional Network (GCN) [8], [13], [14] can mine such potential high-order relationship between the data. GCN transfers the graph structured data to a low-dimensional, compact, and continuous feature space. While

GCN simultaneously has a very successful application in encoding and exploring graph structure and node content, it seems little attention has been given to applying GCNs to deep clustering tasks.

For the purpose of clustering, we can naturally construct an auto-encoder module based on a GCN, the so-called GAE module. GCN can lead the signals to be smoother, which is the inherent advantage of GCN. However, such signal smoothing operation makes the signals more similar, losing the diversity of signals. It has been proven that GCN is prone to over-smoothing when the number of layers becomes large [15], which results in a poor performance in related tasks. So, GCN cannot be stacked as deeply as the CNN model in visual tasks.

To overcome this drawback of GCN, we introduce a common auto-encoder network to supplement the data content information to GAE, like the effect of the residual network. Multiple layers are usually stacked in the deep network, and each layer captures different latent features of the data. To combine the high-order relationship of the data (in GAE) with the potential details of the corresponding content information (in auto-encoder) layer-by-layer, we propose a cross-attention fusion mechanism, which highlights the discriminative information for clustering tasks. Different from the traditional attention mechanism, our cross-attention fusion mechanism fuses two kinds of heterogeneous representations, i.e., the regular data and the irregular graph.

Some datasets naturally contain the relationship graph of samples, e.g., ACM, DLBP and Citeseer. Other datasets come with no such relationship information. We observed an interesting phenomenon. GCN-based methods perform well on the mentioned datasets with relation graph information, while auto-encoder-based methods have the slightly better performance on the ones without relation information. Inspired by this phenomenon, we consider combining the advantages of GCN and auto-encoder to obtain relatively good performance on all datasets.

In this paper, we propose a novel clustering framework, named Cross-Attention Fusion based Enhanced Graph Convolutional Network (CaEGCN). In CaEGCN, we can extract the high-order relationship between the data through the Graph Convolutional Auto-encoder module (GAE). To alleviate the over-smoothing problem of GAE and supplement the content information to GAE, we build a Content Auto-encoder module (CAE) composed of a common auto-encoder, which extracts the content information of the data. Besides, we propose a cross-attention fusion mechanism to encode the above two modules to output a complete representation. In order to guide the optimal clustering direction of the entire model in an end-to-end manner, we introduce a self-supervised module.

The contributions of this paper are listed as follows as a summary,

- We propose an end-to-end cross-attention fusion based deep clustering framework, in which the cross-attention fusion module creatively concatenates the graph convolutional auto-encoder module and content auto-encoder module in multiple layers;
- We propose a cross-attention fusion module to assign the attention weights to the fused heterogeneous representation;
- In the graph convolutional auto-encoder module, we propose simultaneously reconstructing the content and relationship between the data, which effectively strengthens the clustering performance of CaEGCN;
- We test CaEGCN on the natural language, human behavior and image datasets to prove the robustness of CaEGCN.

The rest of the paper is organized as follows. In Section 2, we briefly review the graph convolutional network, deep clustering and attention mechanism, respectively. In Section 3, we detail the cross-attention fusion based enhanced graph convolutional network for clustering by presenting the four main modules. In Section 4, the proposed method is evaluated on clustering problems with several public datasets. Finally, the conclusion and the future work are discussed in Section 5.

2 Related Work

In this section, we review the necessary knowledge related to the research of this paper, which are *graph convolutional network, deep clustering and attention mechanism*.

2.1 Graph Convolutional Network (GCN)

Many research fields consider certain natural graph structures, such as the traffic road network [16], the human skeleton points [17] and molecules structures in biology [18]. Graph is a kind of irregular structural data, which is dispersive and disorderly. To cope with such irregular data, a lot of GCN based methods have been proposed. These methods can be divided into two main categories: spectral-based GCN methods [13], [14], [15] and spatial-based GCN methods [8], [19]. In a GCN, nodes can be assigned to the features of data, and the edge weight information describes the similarity between nodes, which shows that graph has a strong information organization ability.

The spectral-based GCN methods exploit the spectrum representation of a graph. Kipf *et al.* [15] initially proposed the graph convolutional networks for prediction tasks, which simulates the graph convolutional operation through the local first-order approximation of spectral convolutions. Levie *et al.* [20] proposed CayleyNet, which further applies the Cayley polynomials to capture the narrow bands in the frequency domain. Wang *et al.* [21] introduced the generative adversarial mechanism into the learning of graph representation, and developed a new graph softmax function utilizing the latent structure information of the data.

The spatial-based GCN methods directly define the operations on the graph and extract the information from the spatial neighbor groups. Hamilton *et al.* [19] proposed GraphSage, which chooses a fixed number of neighbors for each sample, thereby improving the efficiency of spatial graph convolution. Velickovic *et al.* [8] proposed a graph attention network, which computes the corresponding hidden information for each node and uses the attention mechanism to weight the importance of each node compared with its neighbors. One year later, Velickovic *et al.* [22] introduced the idea of maximizing the mutual information into GCN, driving the local

network embedding to capture the global structural information. More comprehensive reviews about GCN can be found in [23].

2.2 Deep Clustering

The current deep learning researches mainly concentrate on the supervised learning tasks. How to extend it onto a framework for unsupervised clustering is a meaningful problem. Fortunately, some researchers have conducted the related works. Xie *et al.* [2] proposed a deep embedded clustering method, which exploits the deep learning to learn the feature representations and the cluster assignments of the data. Ji *et al.* [3] constructed a self-expression layer between the encoder and decoder of the auto-encoder.

With the development of multi-view clustering, more and more researchers introduce the relationship between the data to enhance the clustering performance. Kipf *et al.* [24] used the graph convolutional encoder and an inner product decoder to build a Variational Graph Auto-encoder (VGAE), which learns the latent features of undirected graphs for clustering. Pan *et al.* [25] improved the VGAE framework and introduced an adversarial regularization rule to optimize the learned representation for clustering. Wang *et al.* [26] employed the graph attention network to weight the importance of neighboring nodes, and obtained a more accurate representation of each node. Bianchi *et al.* [27] continuously relaxed the normalized minCUT problem and trained the graph neural network to obtain the cluster assignments. Zhu *et al.* [28] modified the Markov diffusion kernel to derive a variant of GCN, called simple spectral graph convolution, which obviously reduces the computation cost.

Li *et al.* [29] jointed the advantages of K-means and spectral clustering, and embedded it into the graph auto-encoder to generate the better data representations. Bo *et al.* [30] transferred the representation learned by the auto-encoder to the corresponding GCN, and proposed a dual self-supervision mechanism to unify these two different deep neural architectures, which is an important baseline in this paper.

In the above, GCN-based clustering methods update network parameters by reconstructing the adjacency matrix and sufficiently exploiting the structure information, but they ignore the node information and the over-smoothing problem.

2.3 Attention Mechanism

Recently, in the fields of machine translation [31], [32], semantic segmentation [33] and image generation [34], the attention mechanism has become a trick module that improves the effectiveness of most models. Self-attention is a variant of the traditional attention mechanism. Vaswani *et al.* [31] proposed the self-attention mechanism in machine translation applications, which obtains the satisfactory experimental results. Besides, self-attention mechanism is robust and easily embedded into recurrent neural networks [35], generative adversarial networks [36] and other neural networks, which also achieves the excellent experimental results.

Many scholars continuously optimize the self-attention mechanism. Wang *et al.* [37] introduced a dependency tree into the self-attention mechanism to represent the relationship between words. Yu *et al.* [38] extracted the local information through the convolution model to complement the global interaction of the self-attention mechanism. Xue *et al.* [39] used the self-attention mechanism in image segmentation, which better achieves the accurate segmentation through the long-range context relations. Devlin *et al.* [40] proposed the self-attention-based Bidirectional Encoder Representations (BERT) to more thoroughly capture the bidirectional relationship in sentences, which dramatically refreshes the experimental performance in the NLP field.

Self-attention mechanism also obtains the success in the computer vision field. Dosovitskiy *et al.* [41] further proposed a Vision Transformer that applies a pure transformer directly to the sequences of image patches. Carion *et al.* [42] migrated the transformer to the object detection task, which removes many hand-designed components and realizes the end-to-end automatic training and learning. Huang *et al.* [43] designed a transformer-based network model for the 3D hand pose estimation.

To simultaneously handle the heterogeneous data in the proposed model, i.e., the regular data and the irregular graph, we propose the cross-attention fusion module in this paper.

3 Cross-Attention Fusion Based Enhanced Graph Convolutional Network

In this section, we present a novel cross-attention fusion based enhanced graph convolutional network model, which sufficiently integrates the content information and the relationship between the data in a multi-level adaptive manner to improve the clustering performance.

The overall network architecture shown in Fig. 1 consists of four main modules:

- An *auto-encoder module* extracts the content information, which can be expressed as $\mathbf{H}_l = CAE(\mathbf{H}_{l-1})$;
- A *GCN based auto-encoder module* encodes the relationship between the data to the latent representations, and is expressed as $\mathbf{Z}_l = GAE(\mathbf{R}_{l-1}, \mathbf{A})$;
- A *cross-attention module* concatenates above two modules, i.e., $\mathbf{R}_l = F_{\text{att}}(\gamma\mathbf{Z}_l + (1-\gamma)\mathbf{H}_l)$;
- A *self-supervised module* constrains the consistency of the distributions of middle layer representations.

3.1 Constructing the Graph

Before presenting the proposed CaEGCN model, we first construct the necessary graph of raw data. Given a set of data $\mathbf{X} \in \mathbb{R}^{D\times N}$ containing N samples and the ith sample $\mathbf{x}_i \in \mathbb{R}^D$, we employ the commonly-used K-nearest neighbor (KNN) to construct the corresponding graph to exhibit its structure information.

For the image data, we calculate the similarity between samples using the heat kernel method as [44]

$$s_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{t}}, \tag{1}$$

where t represents the variance scale parameter.

As for the natural language data, the inner-product method is chosen to measure the similarity between samples as follow:

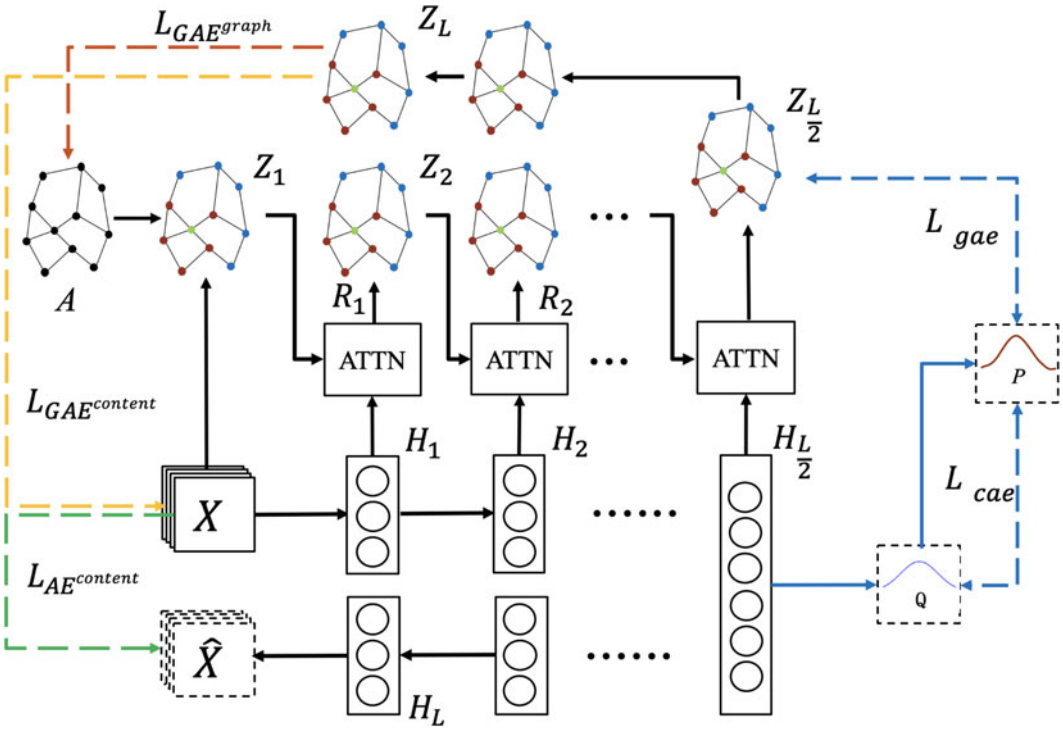


Fig. 1. The conceptual framework of CaEGCN is displayed, which includes four modules: content auto-encoder module, graph convolutional auto-encoder module, cross-attention fusion module and self-supervised module. $\mathbf{X}$ is the original data, $\hat{\mathbf{X}}$ is the reconstructed data, and $\mathbf{A}$ is the original graph. $\mathbf{H}_l$ and $\mathbf{Z}_l$ represent the lth layer output of CAE and GAE, respectively. $\mathbf{R}_l$ is the cross-attention fused representation of $\mathbf{H}_l$ and $\mathbf{Z}_l$. $\mathcal{L}_{CAE^{\text{content}}}$ is the content reconstruction loss of CAE. $\mathcal{L}_{GAE^{\text{content}}}$ and $\mathcal{L}_{GAE^{\text{graph}}}$ are the content reconstruction loss and graph reconstruction loss of GAE, respectively. $\mathcal{L}_{\text{cae}}$ and $\mathcal{L}_{\text{gae}}$ form the self-supervised module losses.

$$s_{ij} = \mathbf{x}_j^T \mathbf{x}_i. \tag{2}$$

Then, with the above calculated similarities among all samples, we pick up the K highest correlation neighbors of each sample and connect them; so, a graph $\mathbf{A}$ is obtained. In many applications, the graph information actually comes with the given dataset $\mathbf{X}$.

3.2 Content Auto-Encoder Module (CAE)

As we know, deep convolutional network can effectively extract the critical features from the complex data. Auto-encoder can reconstruct the samples and reduce the missing information during the learning procedure, which is naturally proper for unsupervised learning. To extract the content information in the data, we first train a deep convolutional network based auto-encoder module, which is named as Content Auto-encoder Module (CAE).

We represent the input of the lth layer as $\mathbf{H}_{l-1}$, then its output $\mathbf{H}_l$ can be obtained by

$$\begin{aligned} \mathbf{H}_l &= CAE(\mathbf{H}_{l-1}) \\ &= \sigma(\mathbf{U}_l \mathbf{H}_{l-1} + \mathbf{b}_l), \quad l = 1, 2, \ldots, L, \end{aligned} \tag{3}$$

where the activation function σ can be chosen according to the practical applications, such as ReLU or Sigmoid. $\mathbf{U}_l$ and $\mathbf{b}_l$ denote the weight and bias of the lth layer of CAE, respectively.

In addition, the input in the first layer of CAE is the raw data $\mathbf{X}$, i.e., $\mathbf{H}_0 = \mathbf{X}$. The output in the final layer reconstructs the raw data, i.e., $\hat{\mathbf{X}} = \mathbf{H}_L$, and the final loss function of CAE can be defined as

$$\mathcal{L}_{CAE^{\text{content}}} = \frac{1}{2} ||\mathbf{X} - \hat{\mathbf{X}}||_F^2, \tag{4}$$

where $\| \cdot \|_F^2$ denotes the Frobenius norm.

3.3 Cross-Attention Fusion Module

As shown in Fig. 1, CAE extracts the content information in the data, and GAE exploits the corresponding relationship between the data. How to fuse these two kinds of information for clustering tasks is a key problem.

Cross-attention fusion mechanism has the global learning ability and good parallelism, which can further highlight the critical information in the fusion representations while suppressing the useless noise. Therefore, we use the cross-attention fusion mechanism to integrate the content information learned by CAE and the data relationship learned by GAE in a multi-level adaptive manner, which is the so-called Cross-Attention Fusion Module.

We define the cross-attention fusion mechanism as

$$\mathbf{R} = F_{\text{att}}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = F_{\text{att}}(\mathbf{W}^q \mathbf{Y}, \mathbf{W}^k \mathbf{Y}, \mathbf{W}^v \mathbf{Y}), \tag{5}$$

where $\mathbf{R}$ is the cross-attention fusion representation of the overall data. $\mathbf{W}^q$, $\mathbf{W}^k$ and $\mathbf{W}^v$ are three transformation matrices. In other words, we get the query $\mathbf{Q} = \mathbf{W}^q \mathbf{Y}$, the key $\mathbf{K} = \mathbf{W}^k \mathbf{Y}$ and the value $\mathbf{V} = \mathbf{W}^v \mathbf{Y}$.

The raw fusion representation $\mathbf{Y}$ is the input of the cross-attention fusion module, which is defined as

$$\mathbf{Y} = \gamma \mathbf{Z}_l + (1 - \gamma) \mathbf{H}_l, \tag{6}$$

where $\mathbf{H}_l$ is the output of lth layer in CAE and $\mathbf{Z}_l$ is the output of corresponding layer in GAE (shown in formula (12)).

γ is a trade-off parameter to balance the importance of $\mathbf{Z}_l$ and $\mathbf{H}_l$, which is set to 0.5 in our experiments.

To discover the latent relationship between data and generalize the cross-attention fusion mechanism $F_{\text{att}}(\cdot)$ in (5), we first calculate the similarity s_{ij} between the fusion query $\mathbf{q}_i$ and the fusion key $\mathbf{k}_j$

$$s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j, \tag{7}$$

where $\mathbf{q}_i$ and $\mathbf{k}_j$ denote the ith and jth vectors in $\mathbf{Q}$ and $\mathbf{K}$, respectively.

Then, we execute the softmax normalization on above $\mathbf{s}_{ij}$ to obtain the relevance weight α_{ij} as follows:

$$\alpha_{ij} = \text{softmax}(s_{ij}) = \frac{\exp(s_{ij})}{\sum_{j=0}^{N} \exp(s_{ij})}. \tag{8}$$

Finally, the output of the cross-attention fusion mechanism $\mathbf{R} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_N)$, i.e., the fusion representation of the data content information and the relationship between data, can be written as

$$\mathbf{r}_i = \sum_{j=0}^{N} \alpha_{ij} \mathbf{v}_j, \quad i = 1, 2, \ldots, N, \tag{9}$$

where $\mathbf{r}_i$ is the ith sample's cross-attention fusion representation.

To further perceive different aspects of the data, multi-head mechanism is also introduced, which contains multiple parallel cross-attention fusion modules. Specifically, we repeatedly project the query $\mathbf{Q}$, key $\mathbf{K}$ and value $\mathbf{V}$ to obtain M parallel cross-attention modules. Each cross-attention fusion module is regarded as one head, and each head has the different weight matrices $\{\mathbf{W}_m^q \in \mathbb{R}^{N \times D_l}, \mathbf{W}_m^k \in \mathbb{R}^{N \times D_l}, \mathbf{W}_m^v \in \mathbb{R}^{N \times D_l}\}$ to linearly transform the fusion features $\mathbf{Q}_m = \mathbf{W}_m^q \mathbf{Q}$, $\mathbf{K}_m = \mathbf{W}_m^k \mathbf{K}$, $\mathbf{V}_m = \mathbf{W}_m^v \mathbf{V}$ where D_l is the dimensionality of the lth layer. The mth head is

$$\mathbf{R}^m = F_{\text{att}}(\mathbf{Q}_m, \mathbf{K}_m, \mathbf{V}_m), m = 1, 2, \ldots, M. \tag{10}$$

We concatenate the outputs of all M heads, and multiply the weight matrix $\mathbf{W} \in \mathbb{R}^{N \times (M \times D_l)}$ to get the final cross-attention fusion representation

$$\mathbf{R} = \mathbf{W} \cdot \text{Concat}(\mathbf{R}^1, \ldots, \mathbf{R}^M), \tag{11}$$

where $\text{Concat}(\cdot)$ denotes the matrix concatenate operation. This is the so-called multi-head mechanism and cross-attention fusion module.

3.4 Graph Convolutional Auto-Encoder Module (GAE)

As we mentioned before, the relationship between the data can effectively improve the clustering performance. Most deep clustering methods only consider the content information of data, while ignoring the important data relationship [2]. Fortunately, Graph Convolutional Network (GCN) [45] is able to handle such relationships and the content information of data collaboratively. To exploit GCN in unsupervised clustering tasks, we propose a GCN based Auto-Encoder module (GAE), which creatively reconstructs both graph and content information.

The previous cross-attention mechanism module combines the content representation $\mathbf{H}_l$ from CAE with the relationship representation $\mathbf{Z}_l$ from GAE to output a fusion representation $\mathbf{R}_l$ in different layers. Then, the GAE executes the spectral graph convolution on such $\mathbf{R}_l$ to learn the high-order discriminative information based on the adjacency matrix $\mathbf{A}$. Finally, the middle layer $\mathbf{Z}_{\frac{L}{2}}$ is used for clustering.

The convolution operation in each GAE layer can be expressed as follow:

$$\begin{aligned} \mathbf{Z}_l &= GAE(\mathbf{R}_{l-1}, \mathbf{A}) \\ &= \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}} \mathbf{R}_{l-1} \mathbf{U}_l), \end{aligned} \tag{12}$$

where $\hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}}$ is the approximated graph convolutional filter and $\hat{\mathbf{D}}$ is the degree matrix of $\hat{\mathbf{A}}$, where $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$. With the identity matrix $\mathbf{I}$ and the adjacency matrix $\mathbf{A}$, we use $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ to ensure the self-loop in each node. Additionally, $\mathbf{U}_l$ denotes the weight of the lth layer, and $\mathbf{Z}_l$ is the output of the lth GAE layer.

It should be noted that the input of the first layer in GAE is slightly different. The first layer just uses the raw data $\mathbf{X}$ instead of $\mathbf{R}_0$ as input

$$\mathbf{Z}_1 = GAE(\mathbf{X}, \mathbf{A}) = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X} \mathbf{U}_1). \tag{13}$$

After this multi-layer learning, the GAE encoder encodes both the raw relationship A and the content $\mathbf{X}$ into a useful representation $\mathbf{Z}_{\frac{L}{2}}$. In order to preserve more information, we set the graph reconstruction and content reconstruction errors as the loss functions of GAE.

i) Graph Reconstruction Loss. We choose a simple inner product operation to reconstruct the relationship between samples as [24]

$$\tilde{\mathbf{A}} = \text{Sigmoid}(\mathbf{Z}_L^T \mathbf{Z}_L), \tag{14}$$

where $\mathbf{Z}_L$ is the output of the last GAE layer, and $\tilde{\mathbf{A}}$ is the reconstructed adjacency matrix. The loss of graph reconstruction can be defined as

$$\mathcal{L}_{GAE^{\text{graph}}} = ||\mathbf{A} - \tilde{\mathbf{A}}||_F^2. \tag{15}$$

By minimizing the error between $\mathbf{A}$ and $\tilde{\mathbf{A}}$, the GAE module may preserve more data relationship in the latent representation $\mathbf{Z}_{\frac{L}{2}}$ to improve the clustering performance.

ii) Content Reconstruction Loss. Except for the relationship between the data, we also constrain the GAE module to preserve enough content information, which has much difference with formula (4); so, we creatively define its loss function as

$$\mathcal{L}_{GAE^{\text{content}}} = ||\mathbf{X} - \mathbf{Z}_L||_F^2, \tag{16}$$

where $\mathbf{Z}_L$ is the output of the last layer in GAE, which has the same size with the raw data $\mathbf{X}$. In this way, the GAE encodes both the relationship and the content of samples into a discriminative representation for clustering.

3.5 Self-Supervised Module

It is difficult to judge whether the learned representation $\mathbf{Z}_{\frac{L}{2}}$ is optimally for clustering during the optimization procedure. We need to give an optimization target about clustering.

To solve this problem, we first get a set of initial cluster centers $\{\beta_c\}_{c=1}^{C}$ by performing K-means on $\mathbf{H}_{\frac{L}{2}}$, where C is the number of clusters. These cluster centers guide approximately the optimization direction for $\mathbf{Z}_{\frac{L}{2}}$, which is the so-called Self-Supervised Module.

We use the Student's t-distribution [45] to calculate the similarity between the middle layer representation $\mathbf{H}_{\frac{L}{2}}$ and the cluster centers β_c as follow:

$$t_{ic} = \frac{(1 + ||\mathbf{h}_i - \beta_c||^2)^{-1}}{\sum_{c=1}^{C}(1 + ||\mathbf{h}_i - \beta_c||^2)^{-1}}, \tag{17}$$

where $\mathbf{h}_i$ is the ith sample representation in $\mathbf{H}_{\frac{L}{2}}$. And t_{ic} measures the probability that the ith sample is assigned to the cth cluster, so $T = [t_{ic}]$ is the overall soft assignment distribution.

Furthermore, the choice of target distribution directly determines the clustering quality. We believe that the high-confidence assignments in T is reliable and can be used as the target distribution. We raise p_{ic} to highlight the role of high-confidence distribution

$$p_{ic} = \frac{t_{ic}^2/f_c}{\sum_{c=1}^{C}(t_{ic}^2/f_c)}, \tag{18}$$

where $f_c = \sum_i t_{ic}$ is the soft cluster frequency. The distribution of T and P should be close to each other as follow:

$$\mathcal{L}_{\text{cae}} = \text{KL}(P||T) = \sum_i \sum_c p_{ic} \log \frac{p_{ic}}{t_{ic}}. \tag{19}$$

Similarly, it is easy to construct a soft assignment distribution Z for the representation $\mathbf{Z}_{\frac{L}{2}}$, then we can use the target distribution P to supervise the distribution Z as

$$\mathcal{L}_{\text{gae}} = \text{KL}(P||Z) = \sum_i \sum_c p_{ic} \log \frac{p_{ic}}{z_{ic}}. \tag{20}$$

Now, the optimization goals of GAE and CAE are unified into a distribution P, which makes the learned representation more suitable for clustering tasks.

3.6 Overall Loss Function

The overall objective loss function of Cross-Attention Fusion based Enhanced Graph Convolutional Network (CaEGCN) can be summarized as

$$\begin{aligned}\mathcal{L}_{\text{overall}} = {} & \lambda_1 \mathcal{L}_{GAE^{\text{graph}}} + \lambda_2 \mathcal{L}_{GAE^{\text{content}}} \\ & + \lambda_3 \mathcal{L}_{CAE^{\text{content}}} + \lambda_4 \mathcal{L}_{\text{cae}} + \lambda_5 \mathcal{L}_{\text{gae}},\end{aligned} \tag{21}$$

where λ_1, λ_2, λ_3, λ_4 and λ_5 are the hyper-parameters balancing the importance of difference losses. There are five items in the above objective function, including three reconstruction losses and two self-supervision losses, which optimizes the data representations for clustering tasks from different perspectives.

After optimizing the above objective function, the local optimal representation $\mathbf{Z}_{\frac{L}{2}}$ is obtained. Then, we perform the softmax operation on $\mathbf{Z}_{\frac{L}{2}}$ to get the final clustering results, i.e., $\max(\text{softmax}(\mathbf{Z}_{\frac{L}{2}}))$.

4 Experimental

In this section, CaEGAN is evaluated on various type of public datasets, including natural language, human behavior and image datasets. We present the experimental settings and analysis below.

4.1 Datesets

- ACM [30] contains 3025 papers of 3 major categories (i.e., database, wireless communication and data mining). The keywords of each paper are chosen as its feature. Different papers of the same author should have the relative strong correlation, so we can construct the structure graph for GCN.
- DBLP [46] is an author network dataset collected from the DBLP website, which includes 4057 authors of 4 categories. The research fields of each author are treated as the feature.
- Citeseer[1] is a citation network dataset which is composed of paper features and citation connections between papers. This dataset has 3327 papers of 6 categories.
- HHAR [47] consists of 10299 sensor records collected from smart phones and smart watches, which is divided into 6 categories, including biking, sitting, standing, walking, stair up and stair down.
- USPS [48] contains 9298 gray images of 10 handwritten digits, and the size of each images is 16×16.
- COIL-20 [49] collects from 20 objects. Each object is rotated 360 degree horizontally, and a photograph is taken every 5 degree; so, there are 72 gray-scale images for each object and 1440 images in total. The image pixel is 64×64.

4.2 Compared Methods

To verify the effectiveness of CaEGCN, we compare it with several state-of-the-art clustering methods, including,

- *K-means* [48] is a basic clustering algorithm based on the content of data only.
- *Auto-Encoder (AE)* [1] performs K-means on the low-dimensional representations learned from the deep auto-encoder network.
- *Improved deep embedded clustering (IEDC)* [4] adds the clustering oriented loss and the reconstruction loss to the deep auto-encoder network, which realizes the one-step clustering of low-dimensional representations.
- *Variational Graph Auto-Encoders (VGAE)* [24] is a variational graph auto-encoder with both topology and content information, which introduces the GCN architecture and the graph reconstruction loss to build a graph convolutional auto-encoder network.
- *Adversarially Regularized Graph Auto-encoder (ARGA)* [25] is a GAN architecture deep clustering model. They first construct a graph convolutional auto-encoder network, then the adversarial training principle is applied to enforce the latent codes to match a prior Gaussian or uniform distribution.

1. https://csxstatic.ist.psu.edu/downloads/data

TABLE 1
Clustering Results on All Five Datasets

Dataset	Metric	K-means	AE	IEDC	VGAE	ARGA	DAEGC	SDCN	CaEGCN
ACM	ACC	0.6820	0.8278	0.8645	0.8294	0.8327	0.8694	0.8860	**0.9012**
	NMI	0.3263	0.5020	0.5824	0.5285	0.5039	0.5618	0.6326	**0.6703**
	ARI	0.3119	0.5553	0.6421	0.5618	0.5646	0.5935	0.6931	**0.7300**
	F1	0.6846	0.8295	0.8632	0.8286	0.8335	0.8707	0.8857	**0.9009**
DBLP	ACC	0.3646	0.5435	0.6571	0.5763	0.5450	0.6205	0.6613	**0.6823**
	NMI	0.0886	0.2220	0.3080	0.2189	0.2019	0.3249	0.3249	**0.3388**
	ARI	0.0657	0.1651	0.3210	0.2348	0.1949	0.2103	0.3338	**0.3617**
	F1	0.2637	0.5325	0.6439	0.5456	0.5343	0.6175	0.6556	**0.6669**
Citeseer	ACC	0.3384	0.5909	0.6023	0.5161	0.5912	0.6454	0.6222	**0.6802**
	NMI	0.1502	0.3066	0.3074	0.2572	0.3069	0.3641	0.3601	**0.4000**
	ARI	0.0893	0.3134	0.2924	0.2405	0.3138	0.3778	0.3623	**0.4240**
	F1	0.2246	0.5483	0.5230	0.4184	0.5485	**0.6220**	0.5893	0.6138
HHAR	ACC	0.5998	0.4621	0.7920	0.6252	0.7040	0.7651	0.8449	**0.8742**
	NMI	0.5887	0.3610	0.7960	0.6059	0.7154	0.6910	0.8021	**0.8256**
	ARI	0.4609	0.2257	0.7033	0.4601	0.6114	0.6038	0.7292	**0.7627**
	F1	0.5833	0.4182	0.7333	0.5696	0.6667	0.7689	0.8297	**0.8724**
USPS	ACC	0.6682	0.4402	0.7684	0.6381	0.7196	0.7355	0.7722	**0.7755**
	NMI	0.6272	0.4850	0.7795	0.7004	0.6859	0.7112	0.7907	**0.7923**
	ARI	0.5464	0.3082	0.7011	0.5636	0.6081	0.6333	**0.7110**	0.7107
	F1	0.6494	0.3665	0.7565	0.5861	0.7093	0.7245	0.7626	**0.7634**
COIL-20	ACC	0.5271	0.6181	0.6354	0.6188	0.6590	0.6375	0.6674	**0.6989**
	NMI	0.6890	0.6760	0.7626	0.7494	0.7440	0.7672	0.7775	**0.8266**
	ARI	0.4542	0.4918	0.5792	0.5678	0.5673	0.5845	0.6052	**0.6803**
	F1	0.4947	0.6030	0.6113	0.5937	0.6327	0.6042	0.6428	**0.6845**

We mark the best-performing and the second-best-performing results by bolded and underlined.

- *Deep Attentional Embedded Graph Clustering (DAEGC)* [26] uses the graph attention network to build the encoder and trains the internal product decoder to reconstruct the graph structure. In addition, soft labels are generated according to the graph embedding to monitor the self-training graph clustering process.
- *Structural Deep Clustering Network (SDCN)* [30] uses the structure information learned by the GCN module to strengthen the data representation learned by the auto-encoder. Furthermore, it constructs a dual self-supervised loss to combine two networks and supervise clustering, which is an important baseline.

The parameter settings of compared methods are listed below. For K-means, we repeatly run K-means 20 times and report the best result. For AE and IEDC, following the work in [30], the network dimension of each dataset is set as $500-500-2000-10-2000-500-500$. VGAE is a two-layer network, so we set network dimension of its encoder as $500-10$, [24]. For ARGA, following the work in [25], we set the dimension of the encoder as $32-16$, in addition, the discriminator's dimension is set as $16-64$. For DAEGC, the dimension of the graph attention encoder is set as $256-16$, [26]. For SDCN, we set the dimensions of the encoder and GCN module for $500-500-2000-10$, [30].

To evaluate all methods from multiple aspects, we choose four popular clustering evaluation metrics, including Accuracy (ACC), Normalized Mutual Information (NMI), Average Rand Index (ARI) and macro F1-score (F1). For all metrics, a higher score indicates a better clustering quality.

4.3 Parameter Settings

When a dataset does not come with graph information, to construct the initial graph of data, we select the popular K-Nearest-Neighbor algorithm (KNN), and the value of K is positively correlated with the number of samples and categories. Following the strategy in SDCN [30], we tune different K in the range {3,5,10} to get the best performance. Generally, USPS and HHAR employ $K=10$ and $K=5$ to construct the corresponding graphs, respectively. As for ACM, DBLP, and Citeseer, we directly exploit the existing graphs in the datasets.

In the proposed CaEGCN, we set the dimensions of both CAE and GAE modules as $input-500-10-cluster-500-500-output$, where $input$ and $output$ denote the dimension of the raw data, and $cluster$ represents the number of cluster categories. The purpose of the last layer in the decoder is to reconstruct the raw data, so the dimension of the last layer equals to the first layer, i.e., $output=input$. We set the hyper-parameters of the objective function as $\lambda_1=0.01$, $\lambda_2=0.01$, $\lambda_3=1$, $\lambda_4=0.1$, $\lambda_5=0.01$.

In the cross-attention fusion module, we set the number of heads as 8. In the self-supervised module, the iteration number of K-means is set as 1000 to initialize the cluster centers. At last, we employ the Xavier method to initialize our model parameters [50], and the initial learning rate is set as 0.001.

4.4 Experiment Results Analysis

Table 1 exhibits the whole experimental results compared with other clustering methods. Obviously, the proposed

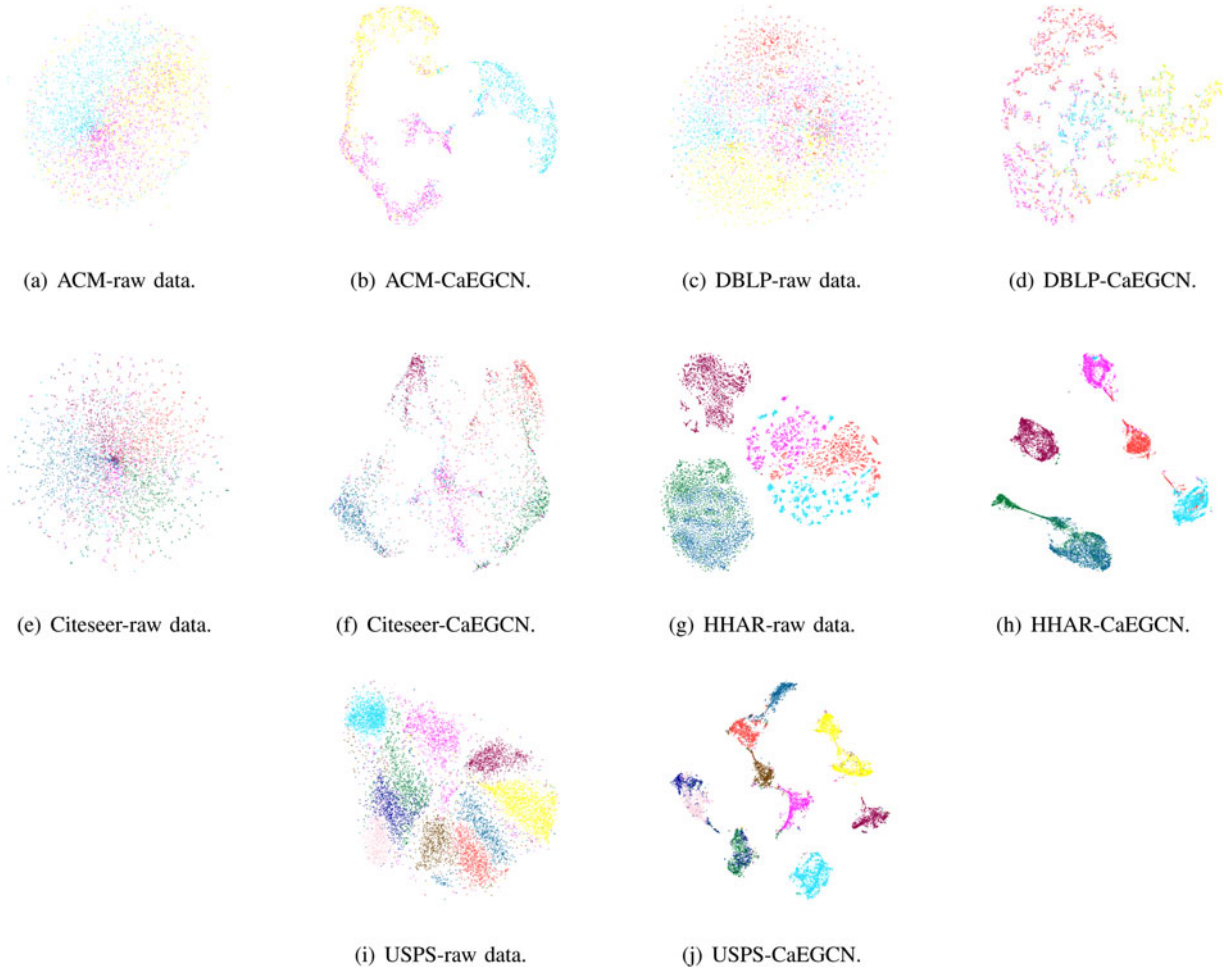


Fig. 2. 2D visualization. The comparison of the raw data and the clustering results of CaEGCN on ACM, DBLP, Citeseer, HAR and USPS datasets.

CaEGCN model achieves the best performance in most cases.

We can see that the content information based deep clustering methods (such as AE and IEDC) work better than the graph convolutional auto-encoder method VGAE. The reason is that VGAE exists the over-smoothing problem. In other words, the information received by the nodes has a low signal-to-noise ratio. SDCN and the proposed CaEGCN supplement the content information into the GCN module in each layer, which effectively relieves the over-smoothing problem, so SDCN and CaEGCN receive the satisfactory performance.

The experimental results show that SDCN and the CaEGCN are superior to other methods on the whole datasets. Compared with directly using GCN, supplementing the content information into the structure representation layer by layer can help the clustering work better, and it also illustrates the significance of the interaction between the heterogeneous information. In addition, the proposed CaEGCN performs better than SDCN in most cases, which proves that the cross-attention fusion module in CaEGCN can promote the learned data representation containing more prominent information for clustering tasks.

For the academic papers datasets, various factors interfere the clustering or recognition tasks, such as the cross-domain applications of popular algorithms, different research topics in the same field and different research fields of the same author, and so on. As for the ACM dataset, the CaEGCN achieves the significant improvements in all four evaluation metrics. The accuracy of CaEGCN increases by from 1.7% compared with SDCN to 24.1% compared with K-means; The NMI of CaEGCN increases from 5.6% compared with SDCN to 51.9% compared with K-means; The ARI of CaEGCN increases by from 5% compared with SDCN to 57.3% compared with K-means; The F1 score of CaEGCN increases by from 1.7% compared with SDCN to 24% compared with K-means.

We also note that VGAE has achieved good clustering results, which simultaneously considers the graph topology information and node content of the graph. ARGA uses the adversarial training to optimize the method based on graph convolution, and achieves some improvements. DAEDC uses the graph attention module, and also achieves the better experimental results. The huge gap between the CaEGCN and SDCN (and others) further proves the superiority of the CaEGCN. Similar to the ACM dataset, DBLP and Citeseer are datasets related to academic papers, and their experimental results show the same pattern and trend.

Large scale dataset is an important challenge for clustering methods. When the scale of samples increases, the performance of many state-of-the-art clustering methods drops dramatically. The scale of HHAR and USPS datasets are 3 times larger than the previous datasets.

TABLE 2
The Results of Ablation Experiments on All Five Datasets

Dataset	Metric	CaEGCN	CaEGCN w/o attention	CaEGCN w/o graph	CaEGCN w/o content
ACM	ACC	**0.9012**	0.8883	0.8860	0.8869
	NMI	**0.6703**	0.6415	0.6391	0.6394
	ARI	**0.7300**	0.6994	0.6989	0.6961
	F1	**0.9009**	0.8879	0.8877	0.8866
DBLP	ACC	**0.6823**	0.6766	0.6628	0.6732
	NMI	**0.3388**	0.3261	0.3127	0.3249
	ARI	**0.3617**	0.3515	0.3257	0.3422
	F1	**0.6669**	0.6581	0.6530	0.6620
Citeseer	ACC	**0.6802**	0.6411	0.6616	0.6267
	NMI	**0.4000**	0.3550	0.3905	0.3533
	ARI	**0.4240**	0.3706	0.3898	0.3463
	F1	**0.6138**	0.5975	0.5744	0.5406
HHAR	ACC	**0.8742**	0.8637	0.8693	0.8650
	NMI	**0.8256**	0.8090	0.8231	0.8224
	ARI	**0.7627**	0.7320	0.7599	0.7559
	F1	**0.8724**	0.8595	0.8624	0.8568
USPS	ACC	**0.7755**	0.7623	0.7698	0.7704
	NMI	**0.7871**	0.7712	0.7746	0.7794
	ARI	**0.7107**	0.6902	0.6984	0.7019
	F1	**0.7634**	0.7552	0.7566	0.7578
COIL-20	ACC	**0.6989**	0.6794	0.6625	0.6736
	NMI	**0.8266**	0.7957	0.7850	0.7850
	ARI	**0.6803**	0.6362	0.6185	0.6168
	F1	**0.6845**	0.6526	0.6415	0.6457

We mark the best-performing result by bolded.

For the HHAR dataset, it is difficult to distinguish some human daily behaviors, e.g., walking and biking. This imposes a challenge for clustering tasks. Compared with VGAE which integrates the structural features into the content information, the accuracy of the CaEGCN increases by 28.48%; compared with ARGA, the accuracy increases by 19.47%; and compared with DAEGC, the accuracy increases by 12.48%. It is our belief that the poor performance of these methods is due to the over-smoothing problem of GCN. The experimental results prove the effectiveness of our cross-attention module. Compared with the best baseline SDCN, the accuracy of CaEGCN still increases by 3.4%.

For the USPS dataset, its background is simple, so we regard it as a baseline data to test the robustness of the CaEGCN. The experimental results of the CaEGCN, SDCN, and IDEC are similar. This may be caused by the fact that the content information of the some handwritten digit images is difficult to distinct. We emphasize here that each node in the initial graph we constructed can only connect 10 closest nodes when considering efficiency, so the initial graph fails to contain enough valuable relationship between the data. Such a limited graph restricts the learning ability of the convolutional network.

For ACM, DBLP and Citeseer datasets coming with the original graphs, GCN-based methods (e.g., VGAE, ARGA, and DAEGC) achieve better clustering results than the methods based on auto-encoder (e.g., K-means, AE, and IEDC) in most conditions. As for HHAR and USPS datasets that exploit KNN to construct the graphs, the methods based on auto-encoder receive the relatively better performance. Therefore, when the graph accurately describes the relationship between samples, the GCN-based methods can learn the more discriminative information to complete the clustering tasks. When the quality of the graph is not good, the performance of the GCN-based methods are hindered.

The classic SDCN [30] contains the paralled auto-encoder and GCN modules, which achieves sub-optimal clustering results on the ACM, DBLP, HHAR and USPS datasets. Obviously, the fusion operation in SDCN simply sums two data representations extracted by above two modules, neglecting the sample relationship in the fusion representation. Our CaEGCN adaptively fuses such two data representations through a cross-attention mechanism and gets the best clustering results on all datasets.

In summary, these performance improvements of the CaEGCN can be attributed to two aspects: first, the cross-attention fusion mechanism integrates the content information and the data relationship, and highlights the critical information in the fusion representations; second, the self-supervised module further optimizes the distributions of

the middle layer representations to strengthen the performance of deep clustering.

4.5 Clustering Result Visualization

In Fig. 2, we visualize the clustering results of five datasets in a two-dimensional space with the t-SNE algorithm [45]. The location distribution of the raw data is overlapping, while the CaeGCN can obviously drive the raw data into different groups.

4.6 Ablation Experiment Analysis

To prove the effectiveness of each critical module in our model, we designed a set of ablation experiments. Specifically, we repeatedly remove one module from the CaEGCN model and test the clustering performance of these incomplete models on five datasets. The designed incomplete models are displayed as follow,

- *CaEGCN w/o attention:* The proposed CaEGCN without the cross-attention fusion module.
- *CaEGCN w/o graph:* The proposed CaEGCN without the graph reconstruction loss $\mathcal{L}_{GAE^{\text{graph}}}$ in the GAE module

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{GAE^{\text{content}}} + \mathcal{L}_{CAE^{\text{content}}} + \mathcal{L}_{\text{cae}} + \mathcal{L}_{\text{gae}}. \tag{22}$$

- *CaEGCN w/o content:* The proposed CaEGCN without the content reconstruction loss $\mathcal{L}_{GAE^{\text{content}}}$ in the GAE module

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{GAE^{\text{graph}}} + \mathcal{L}_{CAE^{\text{content}}} + \mathcal{L}_{\text{cae}} + \mathcal{L}_{\text{gae}}. \tag{23}$$

From Table 2, we observe that the CaEGCN still achieves the best results on all datasets. The clustering results of the above three incomplete models decline, which verifies the importance of each critical module.

Among them, the experimental results of *CaEGCN w/o attention* drop sharply, which proves that the lack of the content and data relationship fusion can decrease the learning ability of the GAE module.

Without the graph reconstruction loss $\mathcal{L}_{GAE^{\text{graph}}}$, the clustering performance of *CaEGCN w/o graph* still decrease obviously. This reconstruction loss ensures the learned middle layer representations have abundant structure information, which improves the clustering performance. Meanwhile, the experimental results of the *CaEGCN w/o graph* on the datasets using the original graph (e.g., ACM, DBLP) significantly decrese, which reflects that the graph reconstruction loss can effectively improve the quality of data representations with the accurate graph structure.

As for *CaEGCN w/o content*, its experimental results are not bad. Compared with other two modules, the impact of content reconstruction loss $\mathcal{L}_{GAE^{\text{content}}}$ is relatively small. However, we believe that content reconstruction loss is also indispensable. The CAE module likes a residual network to supplement the high-quality content information to the GAE module layer-by-layer. Then, the content reconstruction loss ensures the middle layer representation learned by the GAE module contains more content information of the raw data.

TABLE 3
The Clustering Performance Without the Help of KNN

Dataset	Metric	CaEGCN	CaEGCN w/oKNN
HHAR	ACC	**0.8742**	0.8566
	NMI	**0.8256**	0.7847
	ARI	**0.7627**	0.7316
	F1	**0.8724**	0.8486
USPS	ACC	**0.7755**	0.7322
	NMI	**0.7871**	0.6994
	ARI	**0.7107**	0.6253
	F1	**0.7634**	0.7164
COIL-20	ACC	**0.6989**	0.6604
	NMI	**0.8266**	0.7341
	ARI	**0.6803**	0.5528
	F1	**0.6845**	0.6249

Throughout these three ablation experiments, it turns out that each module improves the clustering performance from different aspects and is meaningful.

4.7 Testing the Effect of KNN

KNN aims to construct a sparse graph based on the fully-connected graph in the proposed model. To prove the effectiveness of KNN, we further conduct an ablation experiment to compare the clustering performance of the sparse graph and the full-connection graph. We call such variant that does not use the sparse graph as *CaEGCN w/o KNN*.

From Table 3, the clustering performance of *CaEGCN* is obviously better than that of *CaEGCN w/o KNN*. We owe this promotion to that KNN preserves the neighbor relationships in the similarity matrix and discards the corresponding irrelevant relationships, which decreases the negative effects of noise and outliers. Therefore, this ablation experiment proves the effectiveness of KNN in the proposed model.

5 Conclusion

In this paper, we proposed a novel end-to-end clustering model, which mainly contains four modules, i.e., content auto-encoder module (CAE), cross-attention fusion module, graph convolutional auto-encoder module (GAE) and self-supervised module. The cross-attention fusion module creatively mines the relationship between samples by fusing the data representations of the CAE and GAE modules layer-by-layer, which largely avoids the over-smoothing problem of GAE. The self-supervised module guides the optimization procedure to improve the clustering performance. In addition, the graph reconstruction loss and content reconstruction loss maintain the sufficient original information in the learned data representations. The excellent experimental results on various datasets prove the superiority of the proposed method.

Supplementary Materials

The supplementary code is available at https://github.com/huogy/CaEGCN.

APPENDIX A
THE SUPERVISED AND SEMI-SUPERVISED LEARNING APPLICATIONS

The propose model can be extended into the supervised and semi-supervised learning applications. However, the self-supervised module for the unsupervised learning has to be removed and the loss function needs to be changed.

A.1 Supervised Learning

Supervised learning is called the inductive learning in the representation learning tasks. We split the dataset into a training set $\mathbf{X}_{\text{train}}$ and a testing set $\mathbf{X}_{\text{test}}$.

(1) *In the training phase,* the CAE network is defined as

$$\mathbf{H}_l = CAE(\mathbf{H}_{l-1}) = \sigma(\mathbf{U}_l\mathbf{H}_{l-1} + \mathbf{b}_l), \tag{24}$$

where the input data in the first layer of CAE is the training set $\mathbf{X}_{\text{train}}$, i.e., $\mathbf{H}_0 = \mathbf{X}_{\text{train}}$. And the GAE network is written as

$$\mathbf{Z}_l = GAE(\mathbf{R}_{l-1}, \mathbf{A}_{\text{train}}) = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}_{\text{train}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{R}_{l-1}\mathbf{U}_l), \tag{25}$$

where its first layer uses the training set $\mathbf{X}_{\text{train}}$ instead of $\mathbf{R}_0$ as the input data, and $\mathbf{A}_{\text{train}}$ is the graph of the training set. The latter layer representation $\mathbf{R}_l$ is the cross-attention fusion representation as

$$\mathbf{R}_l = F_{\text{att}}(\gamma\mathbf{Z}_l + (1-\gamma)\mathbf{H}_l).$$

In addition, a supervised loss function is designed to replace the previous self-supervised loss as follow:

$$\begin{aligned} \mathcal{L}_{GAE^{\text{supervised}}} &= \text{KL}(Y||Z) \\ \mathcal{L}_{CAE^{\text{supervised}}} &= \text{KL}(Y||H), \end{aligned} \tag{26}$$

where Y denotes the distribution of labels of the training set. Z and H are the soft assignment distributions constructed by

$$\begin{aligned} Z &= \max(\text{softmax}(\mathbf{Z}_{\frac{L}{2}})) \\ H &= \max(\text{softmax}(\mathbf{H}_{\frac{L}{2}})). \end{aligned}$$

In above, the overall objective loss function can be summarized as

$$\begin{aligned} \mathcal{L}_{\text{overall}} = \lambda_1\mathcal{L}_{GAE^{\text{graph}}} + \lambda_2\mathcal{L}_{GAE^{\text{supervised}}} \\ + \lambda_3\mathcal{L}_{CAE^{\text{supervised}}}. \end{aligned}$$

(2) *In the testing phase*, the network structure has no change. The input data $\mathbf{X}_{\text{train}}$ and the graph $\mathbf{A}_{\text{train}}$ in formulas (24) and (25) should be replaced by $\mathbf{X}_{\text{test}}$ and $\mathbf{A}_{\text{test}}$.

A.2 Semi-Supervised Learning

Semi-supervised learning contains the inductive semi-supervised learning and the transductive semi-supervised learning in the tasks of representation learning. We divide the dataset into the labeled training data $\mathbf{X}_{label}$, the unlabeled training data $\mathbf{X}_{unlabeled}$ and the testing data $\mathbf{X}_{\text{test}}$.

(1) *Inductive semi-supervised learning* is trained on the labeled data $\mathbf{X}_{label}$ and the unlabeled data $\mathbf{X}_{unlabeled}$, and tested on the testing data $\mathbf{X}_{\text{test}}$. The loss function is defined

$$\begin{aligned} \mathcal{L}_{GAE^{\text{supervised}}} &= \text{KL}(Y||Z) \\ \mathcal{L}_{CAE^{\text{supervised}}} &= \text{KL}(Y||H), \end{aligned} \tag{27}$$

where Y is the distribution of the labeled training data, Z and H are the soft assignment distributions of the corresponding data.

(2) *Transductive semi-supervised learning* is trained on the whole data containing the labeled training data $\mathbf{X}_{label}$, the unlabeled training data $\mathbf{X}_{unlabeled}$, and the testing data without labels $\mathbf{X}_{\text{test}}$. The same loss function (27) is also executed on the labeled training data $\mathbf{X}_{label}$.

REFERENCES

[1] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[2] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[3] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. D. Reid, "Deep subspace clustering networks," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 23–32.

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

[5] J. B. Macqueen, "Some methods for classification and analysis of multivariate observations," *5th Berkeley Symp. Math. Stat Probability*, vol. 1967, pp. 281–297, 1967.

[6] S. S. Tabatabaei, M. Coates, and M. Rabbat, "GANC: Greedy agglomerative normalized cut for graph clustering," *Pattern Recognit.*, vol. 45, no. 2, pp. 831–843, 2012.

[7] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, 2002.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[9] J. Zhang, F. Chen, Y. Guo, and X. Li, "Multi-graph convolutional network for short-term passenger flow forecasting in urban rail transit," *IET Intell. Transport Syst.*, vol. 14, no. 10, pp. 1210–1217, 2020.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, pp. 701–710, 2014.

[11] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

[12] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[14] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 1–14.

[15] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.

[16] L. Zhao *et al.*, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

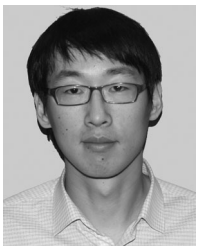
[17] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition." in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 7444–7452.

[18] S. Sanyal, I. Anishchenko, A. Dagar, D. Baker, and P. Talukdar, "ProteinGCN: Protein model quality assessment using graph convolutional networks," *BioRxiv*, pp. 1–11, 2020. [Online]. Available: https://academic.microsoft.com/paper/3015047409, doi: 10.1101/2020.04.06.028266.
[19] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[20] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.
[21] H. Wang *et al.*, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2508–2515.
[22] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.
[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[24] T. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS Workshop on Bayesian Deep Learn.*, 2016, pp. 1–3.
[25] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 50, no. 6, pp. 2475–2487, Jun. 2020.
[26] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 3670–3676.
[27] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 874–883.
[28] H. Zhu and P. Koniusz, "Simple spectral graph convolution," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–15.
[29] X. Li, H. Zhang, and R. Zhang, "Embedding graph auto-encoder with joint clustering via adjacency sharing," 2020, *arXiv:2002.08643*.
[30] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *Proc. Web Conf.*, 2020, pp. 1400–1410.
[31] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[32] G. Tang, M. Müller, A. R. Gonzales, and R. Sennrich, "Why self-attention? A targeted evaluation of neural machine translation architectures," in *Proc. Empirical Methods Nat. Lang. Process.*, 2018, pp. 4263–4272.
[33] J. Fu *et al.*, "Dual attention network for scene segmentation," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3146–3154.
[34] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.
[35] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2016. pp. 551–561.
[36] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7354–7363.
[37] X. Wang, Z. Tu, L. Wang, and S. Shi, "Self-attention with structural position representations," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2019.
[38] A. Yu *et al.*, "QANet: Combining local convolution with global self-attention for reading comprehension," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.
[39] H. Xue, C. Liu, F. Wan, J. Jiao, X. Ji, and Q. Ye, "DANet: Divergent activation for weakly supervised object localization," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6588–6597.
[40] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2018, pp. 4171–4186.
[41] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–22.
[42] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.
[43] L. Huang, J. Tan, J. Liu, and J. Yuan, "Hand-transformer: Non-autoregressive structured modeling for 3d hand pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 17–33.
[44] A. Grigor'yan, *Heat Kernel and Analysis on Manifolds*. Providence, RI, USA: American Mathematical Society / International Press, 2012.
[45] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.
[46] M. Ley, "DBLP: some lessons learned," *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1493–1500, 2009.
[47] A. Stisen *et al.*, "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 127–140.
[48] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[49] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (COIL-20)," *Technical Rep.*, CUCS-005–96, 1996.
[50] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

Guangyu Huo received the BSc degree in IoT engineering and the MS degree in computer science from the Beijing University of Technology, China, in 2016 and 2019, where he is currently working toward the PhD degree in control science and engineering. His research interests include intelligent transportation, computer vision, pattern recognition, and deep learning.

Yong Zhang (Member, IEEE) received the PhD degree in computer science from the BJUT, in 2010. He is currently an associate professor in computer science with BJUT. His research interests include intelligent transportation system, big data analysis, and visualization, computer graphics.

Junbin Gao received the graduation degree from the Huazhong University of Science and Technology (HUST), China, in 1982, the BSc degree in computational mathematics, and the PhD degree from the Dalian University of Technology, China, in 1991. He is currently a professor of big data analytics with the University of Sydney Business School, University of Sydney and was a Pprofessor of computer science with the School of Computing and Mathematics, Charles Sturt University, Australia. He was a senior lecturer and a lecturer in computer science from 2001 to 2005 with the University of New England, Australia. From 1982 to 2001 he was an associate lecturer, lecturer, associate professor, and professor with the Department of Mathematics, HUST. His research interests include machine learning, data analytics, Bayesian learning and inference, and image analysis.



Boyue Wang received the BSc degree in computer science from the Hebei University of Technology, China, in 2012 and the PhD degree from the Beijing University of Technology, China, in 2018. He is currently an associate professor with the Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology, Beijing University of Technology, Beijing. His current research interests include computer vision, pattern recognition, manifold learning, and kernel methods.

Yongli Hu received the PhD degree from the Beijing University of Technology in 2005. He is currently a professor with the Faculty of Information Technology, Beijing University of Technology. He is a researcher with the Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology. His research interests include computer graphics, pattern recognition, and multimedia technology.

Baocai Yin received the PhD degree from the Dalian University of Technology in 1993. He is currently a professor with the Faculty of Information Technology, Beijing University of Technology. He is a researcher with the Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology. His research interests cover multimedia, multifunctional perception, virtual reality, and computer graphics. He is a member of China Computer Federation.