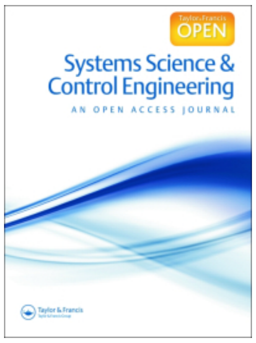

Systems Science & Control Engineering

An Open Access Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tssc20

Deep self-supervised clustering with embedding adjacent graph features

Xiao Jiang, Pengjiang Qian, Yizhang Jiang, Yi Gu & Aiguo Chen

To cite this article: Xiao Jiang, Pengjiang Qian, Yizhang Jiang, Yi Gu & Aiguo Chen (2022) Deep self-supervised clustering with embedding adjacent graph features, Systems Science & Control Engineering, 10:1, 336-346, DOI: 10.1080/21642583.2022.2048321

To link to this article: https://doi.org/10.1080/21642583.2022.2048321

SYSTEMS SCIENCE & CONTROL ENGINEERING: AN OPEN ACCESS JOURNAL
2022, VOL. 10, NO. 1, 336–346
https://doi.org/10.1080/21642583.2022.2048321

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Deep self-supervised clustering with embedding adjacent graph features

Xiao Jiang, Pengjiang Qian, Yizhang Jiang, Yi Gu and Aiguo Chen

School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, People's Republic of China

ABSTRACT

Deep clustering uses neural networks to learn the low-dimensional feature representations suitable for clustering tasks. Numerous studies have shown that learning embedded features and defining the clustering loss properly contribute to better performance. However, most of the existing studies focus on the deep local features and ignore the global spatial characteristics of the original data space. To address this issue, this paper proposes deep self-supervised clustering with embedding adjacent graph features (DSSC-EAGF). The significance of our efforts is three-fold: 1) To obtain the deep representation of the potential global spatial structure, a dedicated adjacent graph matrix is designed and used to train the autoencoder in the original data space; 2) In the deep encoding feature space, the KNN algorithm is used to obtain the virtual clusters for devising a self-supervised learning loss. Then, the reconstruction loss, clustering loss, and self-supervised loss are integrated, and a novel overall loss measurement is proposed for DSSC-EAGF. 3) An inverse-Y-shaped network model is designed to well learn the features of both the local and the global structures of the original data, which greatly improves the clustering performance. The experimental studies prove the superiority of the proposed DSSC-EAGF against a few state-of-the-art deep clustering methods.

ARTICLE HISTORY
Received 16 December 2021
Accepted 26 February 2022

KEYWORDS
Deep clustering; adjacent graph features; self-supervised learning

1. Introduction

Clustering occupies an important position in machine learning (Bickel & Scheffer, 2004; D'Andrade, 1978; Pei & Huang, 2006; Pei & Huang, 2006; Wang et al., 2007; Wang & Huang, 2009; Xianpeng et al., 2019). It is a baseline for computers to recognize things and is widely applied to various applications, such as in gene selection (Pei & Huang, 2006; Pei & Huang, 2006), image segmentation (Xianpeng et al., 2019), and object detection (Wang et al., 2007). The goal of clustering is to partition target data into clusters so that the data instances in the same cluster have similar characteristics and a high degree of similarity. Traditional clustering methods, e.g. hierarchical clustering (D'Andrade, 1978), fuzzy clustering (Yang et al., 2016), K-means clustering (Macqueen, 1967), spectral clustering (Luxburg, 2004), etc., achieve good performance in some applications. However, when the dimension of the original data space is quite high, the similarity measurement among data instances is intractable, which could make clustering invalid or inefficient. To overcome this challenge, raw and high-dimensional data are usually converted into regenerated and low-dimensional features. For this purpose, dimension reduction is commonly used, such as classic Principal Component Analysis (PCA) (Macqueen, 1967). However, traditional dimension reduction methods are sensitive to the noise and outliers involved and usually ignores the nonlinear correlation issues.

Owning to the sustained development of machine learning, deep learning has demonstrated its convincing superiority in quite a few fields recently (Du et al., 2005; Zhao et al., 2004). The fundamental of deep learning is to perform feature extraction layer by layer and to reduce the data size simultaneously. In this sense, a deep neural network (DNN) (Hinton & Salakhutdinov, 2006) can be used for clustering, and this is referred to as deep clustering. As a category of recently emerged clustering methodology, deep clustering derives many practices (Guo et al., 2019; Xie et al., 2016; Yang et al., 2017; Zhang et al., 2018).

Numerous efforts have been made in this regard, but many problems still need to be solved. For example, which type of neural network is suitable for a specific dataset, though there are many neural networks (Jie et al., 2018; Mahardhika et al., 2016; Yu et al., 2018; Yu et al., 2019) available? How to set the loss function appropriately for clustering? What data structure should be retained during the deep learning process?

The basic work of deep clustering focuses on learning to retain certain attributes of the data by adding prior knowledge to the subject. The Deep Embedded

CONTACT Pengjiang Qian qianpjiang@jiangnan.edu.cn

Clustering (DEC) (Xie et al., 2016) is a typical algorithm in this field, which performs feature extraction and clustering at the same time. The algorithm defines an effective objective function called the clustering loss module for the training process. The function can update the parameters of the network and the cluster centre at the same time, and the cluster allocation is implicitly integrated into the soft membership. The subsequent IDEC algorithm (Guo et al., 2019) adds a data recovery process and integrates the reconstruction loss into the loss function, which effectively improves the clustering effect of the algorithm. In (Zhang et al., 2019) the authors creatively use the results of spectral clustering as self-supervision. The spectral clustering result is used as the target distribution to supervise the results of the fully connected network and participate in the update of the parameters of the self-expression layer. Inspired by its ideas, this paper proposes a special self-supervised algorithm. This proposed algorithm is referred to as self-supervised deep clustering with embedded adjacent graph features. To achieve effective clustering, this paper proposes a dedicated model called DSSC-EAGF, which has an improved ability to extract local data features and the data features of global spatial structure.

However, the clustering loss only guarantees the preservation of the local structure, and the similar features of the global structure are ignored. This may lead to the destruction of the credibility of the embedding space. To address this problem, this paper presents the adjacency graph matrix (Yang et al., 2017) to preserve the global structure, and it is embedded into the training process. The method of combining diverse data is also proposed in (Han et al., 2008; Han et al., 2010). In this way, the algorithm allows the encoder to learn local features while learning and retaining global structural features. The major contributions of this paper are summarized as follows.

(1) A dedicated adjacent graph matrix is designed to train the autoencoder in the original data space so that the deep representation of the potential global spatial structure can be obtained.
(2) In the deep encoding feature space, the KNN algorithm is first used to obtain the virtual clusters for devising a self-supervised learning loss. Then, the overall loss measurement of our DSSC-EAGF is realized by integrating the reconstruction loss, clustering loss, and self-supervised loss.
(3) An inverse-Y-shaped network model is designed to well learn the features of both the local and the global structures of the original data, which greatly improves the clustering performance.

The rest of this paper is arranged as follows. Section 2 introduces the related work. Section 3 describes the DSSC-EAGF method and the self-supervised clustering mechanism in detail. The experimental results are analyzed and discussed in Section 4. Section 5 concludes our work and mentions some future work.

2. Related work

2.1. *Deep clustering algorithms*

The current research on deep clustering algorithms mainly has two directions: (1) two-step clustering that first learns an optimal low-dimensional feature representation and then performs clustering on the obtained features; (2) one-step clustering that performs feature learning and clustering at the same time. The feature representation and clustering results are optimized according to the target loss function.

The first type of algorithm mainly employs the existing unsupervised deep learning technology (e.g. autoencoder) to reduce the dimensionality of the original data to remove redundant features. Then, it runs the k-means algorithm on the low-dimensional features to obtain the clustering results. (Yang et al., 2016) applies non-parametric maximum edge clustering to the mid-level awakening clustering of the deep belief network (DBN), where the deep belief network is pre-trained. In (Yang et al., 2017), GR-DNN adds an adjacency graph decoder to train the autoencoder so that the low-level features are obtained, and the information of the neighbouring graphs is preserved. In this way, the acquired features have better similarities and differences, which contributes to better clustering results. Another type of algorithm is to clearly define the clustering loss and use it to guide the clustering. (Yang et al., 2016) proposed a cyclic framework based on depth representation and image clustering. The framework combines the clustering process with feature representation, and it uses three groups of uniformly weighted losses for optimization. DEC (Xie et al., 2016) reduces the data dimension of the original space through deep neural network learning. Meanwhile, it obtains cluster assignment through the clustering module and obtains the feature representation. As an improved version of DEC, the IDEC algorithm (Guo et al., 2019) adds an incomplete self-encoder to maintain the local structure. Compared with DEC, IDEC has better performance on feature extraction and clustering results. However, IDEC has limited performance improvement on the basis of DEC, because it only uses the features of the data itself for clustering, which lacks more prior knowledge. Therefore, this paper combines the above algorithms and adds adjacency graph features for clustering,which achieves good results. We will describe it in detail later.

2.2. Autoencoder module

The autoencoder is essentially a pre-trained neural network. Its target is to achieve the effect of copying the input to the output. The autoencoder has a middle layer h that constructs low-dimension features from the input. Taking the middle layer as the boundary, the neural network can be divided into two parts: the encoder function $h = f_w(x)$ is used to extract features from the input, and the decoder function $x' = g_{w'}(h)$ is used to generate reconstruction results from the features in the middle layer.

Nowadays, two different types of autoencoder are widely used:

(a) Under-complete AutoEncoder. It requires that the h dimension of the middle layer is smaller than the x dimension of the input layer; otherwise, the features cannot be learned. Under this incomplete feature learning, the autoencoder is forced to obtain the most important low-dimensional features of the data.
(b) Denoising AutoEncoder. It does not reconstruct the given x but optimizes the target loss function as shown in (1).

$$\mathrm{L} = \left| |x - g_{w'}(f_w(\widetilde{x}))\right| |_2^2 \tag{1}$$

In Eq. (1), $\widetilde{x}$ is a copy of the original data x that is partially contaminated by noise. In this case, the denoising autoencoder needs to recover data from the contaminated data rather than simply reproduce their input. Thus, the denoising autoencoder prompts the encoder network f_w and decoder network $g_{w'}$ to reduce data dimensionality to generate a distributed structure.

In contrast, our network structure aims to improve the feature extraction of the autoencoder by explicitly strengthening the global structure built based on the degree of global data similarity.

2.3. Deep embedded clustering

The idea of the DEC algorithm (Xie et al., 2016) is to pre-train an autoencoder and then discard the decoder part. The remaining part of the encoder uses the following loss function for fine-tuning:

$$L_c = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2}$$

where q_{ij} is the soft distribution probability between the feature point z_i of the middle layer and the cluster centre μ_j. The loss function is similar to the membership degree in fuzzy clustering (Song et al., 2017), which is calculated through the Student T distribution (Wu et al., 2017):

$$q_{ij} = \frac{\left(1 + ||z_i - \mu_j||^2\right)^{-1}}{\sum_j \left(1 + ||z_i - \mu_j||^2\right)^{-1}} \tag{3}$$

Then, the target distribution defined as follows is used:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \tag{4}$$

It can be seen that the target distribution P is calculated and defined based on Q, so minimizing the loss L_c is a form of self-training (Nigam & Ghani, 2002).

The biggest contribution of DEC (Xie et al., 2016) is the clustering module (or target distribution P). Its working principle is to use a high-confidence target division to guide the training process so that the distribution of the dense samples in each cluster is more concentrated. However, this module also has a defect, that is, it cannot guarantee that the samples near the edge of the cluster will be classified into the correct cluster. To solve this problem, our algorithm explicitly preserves the global structure of the data by using the adjacent graph features. Based on this, the adjacency graph information of the high-confidence samples can effectively promote the samples near the edge to be classified into the correct cluster.

3. Deep self-supervised clustering with embedded adjacent graph features

Our algorithm is essentially an improvement of DEC and IDEC algorithms. It embeds the feature information of the adjacency graph into the deep encoding features to ensure that the acquired features can participate in the clustering. To this end, two sets of neural networks are pre-trained. One is the data autoencoder (DataAE) that takes raw data as the input, and the other is the graph autoencoder (GraphAE) that takes the adjacency graph implying the global spatial information as input. Then, the encoder parts of these two autoencoders are combined so that the data encoder (DataEncoder) can acquire the features of both the adjacent graph and the data.

3.1. Adjacent graph features

The adjacency graph feature can represent the location information of a single data in the global data. The process of dimensionality reduction can only retain the feature information inside the picture, so adding the adjacency graph features can be regarded as artificially adding more features to the dimensionality-reduced data. This can effectively improve the clustering effect. Next, we will describe how to construct adjacency graph features.

Like traditional graph-based methods (e.g. [Belkin & Niyogi, 2003; Hadsell et al., 2007; Hao et al., 2013), the adjacency graph cannot be directly used for large-scale data. If an adjacency graph is constructed on large-scale data directly, the cost of calculating the similarity among all data instances would be greatly increased. Instead, our method is to construct the Anchor-based Graph (Li et al., 2015; Yang et al., 2017). The core idea is to choose a group of data points to approximate the original adjacency graph structure. The chosen data points can represent the original data, and they are called anchor points $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{Na}\}$. Na is set to a value (e.g. $Na = 1000$ in our empirical studies) close to the dimensionality of the input data x (e.g. the resolution of an image: $28 \times 28 = 784$) so that the adjacency graph information would be retained as much as possible. There are a variety of strategies to select the anchor points, such as random selection and using cluster centres obtained by a certain clustering method. The latter is preferred in our method. Compared with random selection, the use of clustering centres (e.g. K-means) is more convenient, and the obtained cluster centres are more representative. For this purpose, the Na anchor points are obtained first through the K-means algorithm with $K = Na$. Then, a similarity matrix$\mathbf{G} \in n \times Na$ between all original data instances $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and the anchor points $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{Na}\}$ is constructed. Specifically, for each $\mathbf{x}_i (i = 1, \ldots, N)$, the similarity between it and its nearest K_1 (e.g. $K_1 = 30$ in our empirical studies) anchor points is set by using the radial basis function (Huang, 2011), and the similarity between x_i and the other points is set to 0. In this way, the adjacency graph matrix $\mathbf{G} \in n \times Na$ is obtained.

3.2. Network

IDEC has shown that autoencoders can retain potential local features of original data during the process of dimensionality reduction. To further integrate the adjacent graph features, this study delicately adds the GraphAE into the proposed model as follows.

First, two complete autoencoders, DataAE (Figure 1(a)) and GraphAE (Figure 1(b)), are pre-trained separately to realize dimensionality reduction and the reconstruction of $\boldsymbol{x}$ and $\mathbf{g} \in \mathbf{G}$. Afterwards, the encoding part of GraphAE is discarded, and the decoder (i.e. GraphDecoder) is kept and integrated into DataAE. That is, any raw data instance $\mathbf{x}$ is input into the DataEncoder of DataAE to obtain the encoding feature denoted as $\mathbf{z}_3$. Then, $\mathbf{z}_3$ is used for GraphDecoder and DataDecoder to obtain the outputs $\mathbf{x}'$ and $\mathbf{g}'$, respectively. In this regard, $\mathbf{z}_3$ can be regarded as a neutralization of $\mathbf{z}_1$ in Figure 1(a) and $\mathbf{z}_2$ in Figure 1(b). As shown in Figure 1(c), the network structure of our method looks like an inverse Y.Next, based on the network structure illustrated in Figure 1(c), our network model needs to be retrained based on a certain loss function, such as.

$$L_r = MSE(\mathbf{x}, \mathbf{x}') + MSE(\mathbf{g}, \mathbf{g}') \tag{5}$$

where $\boldsymbol{x}$ represents the original data; $\boldsymbol{g}$ is the corresponding adjacency graph data; $\mathbf{x}'$ and $\mathbf{g}'$ are the adjacency graph data reconstructed by DataDecoder and GraphDecoder, respectively; $MSE(.)$ is a function to calculate the mean square error.

It should be noted that during this retraining procedure, the network parameters of GraphAE are no longer updated throughout the whole error backpropagation.

When the network illustrated in Figure 1(c) is successfully optimized via the employed loss measurement, a clustering operation can be added behind $\boldsymbol{z}_3$, such as K-means, to perform the clustering task on the encoding features $\mathbf{z}_3$ of all original training data instances. In this way, K_2 (e.g. $K_2 = 10$ on the MNIST dataset [Lecun & Bottou, 1998]. We will discuss the choice of the value of K in a later 4.3.) encoding feature cluster centres can be obtained first. Then, by means of Eq. (3), the soft probability q_{ij} of any encoding feature $\mathbf{z}_{i3} (i = 1, \ldots, n)$ to any encoding feature cluster centre $\boldsymbol{\mu}_j (j = 1, \ldots, K_2)$ can be obtained. After obtaining the required $\mathbf{x}'$, $\mathbf{g}'$ and q_{ij} for the loss function Eq. (2,5), we calculate the loss function according to Eq. (8). The network performs backpropagation according to the loss function to update the network parameters.

3.3. KNN-based self-supervised methodology

As previously mentioned, the soft probabilities between the encoding features and the encoding feature cluster centres can be obtained through Eq. (3). Base on this, our KNN-based self-supervised learning methodology is proposed.

For each $\mathbf{z}_{i3} (i = 1, \ldots, n)$, (The blue solid point in Figure 2), its K_3 (e.g. $K_3 = 7$ in our empirical studies) nearest neighbours (The red solid point in Figure 2) are found and denoted as. $\{\mathbf{z}_{i_1}, \mathbf{z}_{i_2}, \ldots, \mathbf{z}_{i_K_3}\}$For each $\mathbf{z}_{i\_l}$ $(l = 1, \ldots, K_3)$, the soft probabilities $q_{lj} (l = 1, \ldots, K_3; j = 1, \ldots, K_2)$ are calculated by Eq. (3). Afterwards, the so-called referenced mean soft probability q^r_{ij} of $\mathbf{z}_{i3}$ is calculated by Eq. (6):

$$q^r_{ij} = \frac{\sum_l q_{lj}}{K_3} \tag{6}$$

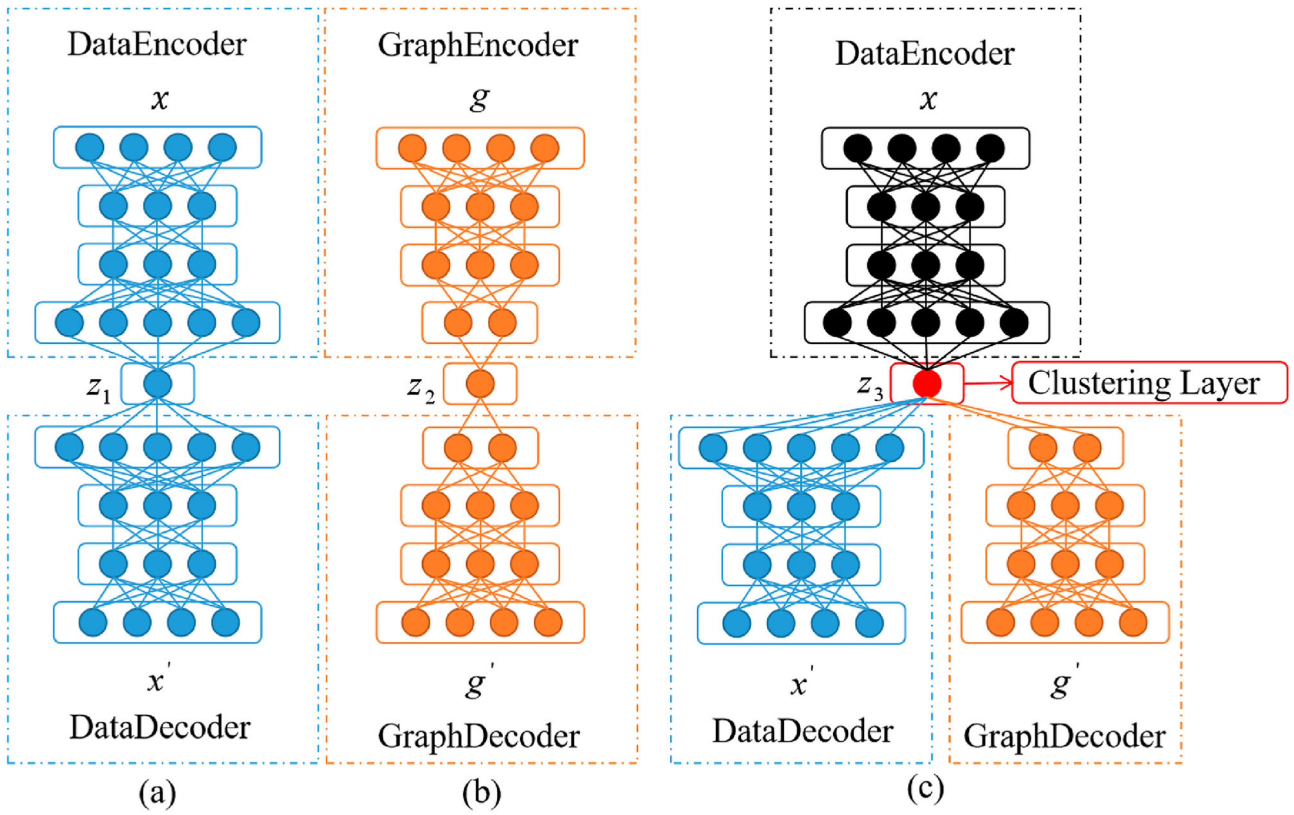

Figure 1. The network structure of DSSC-EAGF. (a), (b), and (c) are the network structures of DataAE, GraphAE, and DSSC-EAGF during pre-training respectively. Among them, (c) is an inverse Y-shape, and the middle layer feature after pre-training, is the input of the Clustering Layer.

Base on Eq. (6), a self-supervised loss measurement is defined as

$$L_{self-supervised} = KL(Q^r||Q) \tag{7}$$

where Q^r consists of q^r_{ij}; Q consists of q_{ij}; $KL(.||.)$ measures the Kullback-Leibler divergence between Q^r and Q.

3.4. Complete loss function integrating self-supervised clustering

By now, our complete loss function that integrates the self-supervised clustering mechanism can be put forward:

$$L = L_r + \lambda L_c + \gamma L_{self-supervised} \tag{8}$$

Where L_r is the reconstruction loss defined in Eq. (5); $L_c$is the clustering loss defined in Eq. (2); $L_{self-supervised}$ is the self-supervised loss defined in Eq. (7); λ and γ are two regularization coefficients and are both set to 0.1, which will be discussed in 4.4.

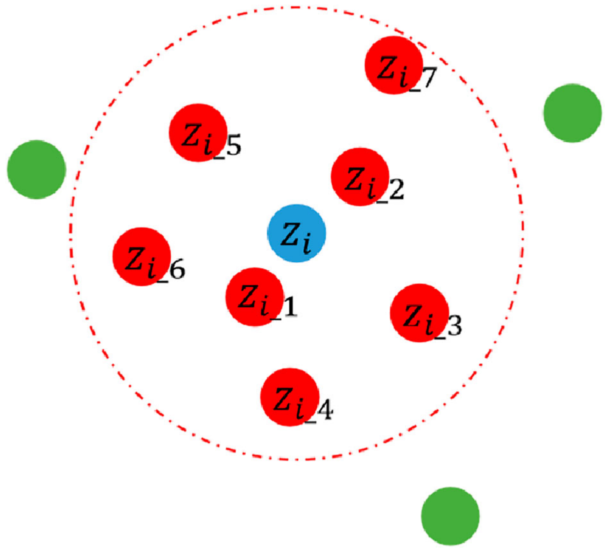

Figure 2. KNN self-supervised module. The blue solid point denotes the current data point $\mathbf{z}_{i3}$ and the red solid denotes the K nearest points.

3.5. Algorithm flow

In this section, our algorithm flow is described. The proposed method is composed of two phases: training and testing (see the two black dashed boxes in Figure 3). The training phase can be further divided into four parts: dataset processing, pre-training (blue dashed box in Figure 3), initialization (green dashed box in Figure 3), and

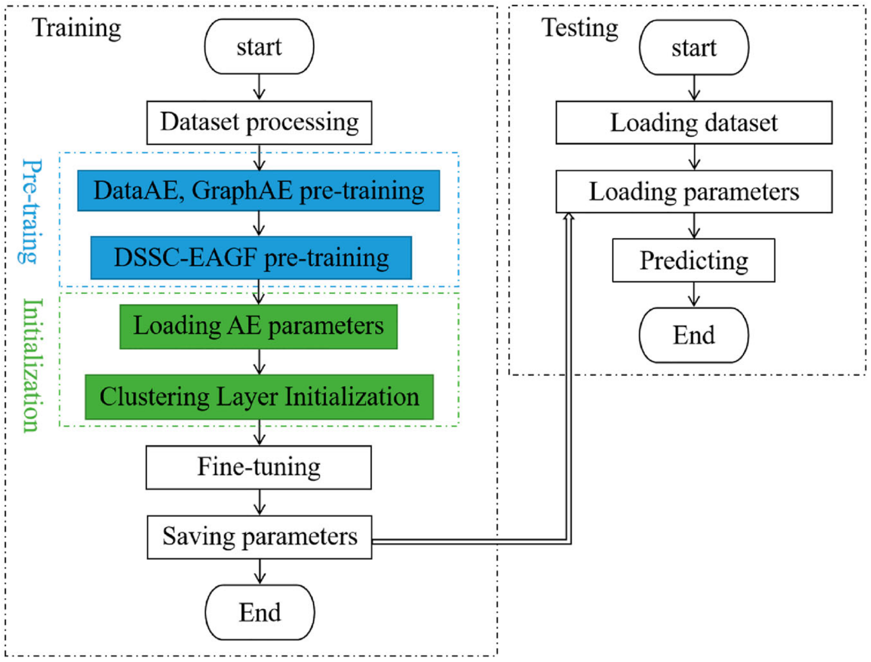

Figure 3. The algorithm flow of DSSC-EAGF.

fine-tuning. In the testing phase, the clustering effect of the trained model is evaluated.

Phase 1-1: Dataset processing. According to Section 3.1, the adjacency graph matrix $\mathbf{G} \in n \times Na$ is initialized using the given training data$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, and $\boldsymbol{g}_i$ matches $\boldsymbol{x}_i$.

Phase 1-2: Pre-training. Using the training data, both DataAE and GraphAE are pre-trained first. Then, based on the mechanism introduced in Section 3.2, the inverse Y-shaped network structure shown in Figure 1(c) is constructed and pre-trained.

Phase 1-3: Initialization. The parameters of the inverse Y-shaped network are loaded into our network model for initialization. Meanwhile, K-means clustering is adopted to obtain the encoding feature cluster centres $\{\boldsymbol{\mu}_j\}_{j=1}^{K_2}$, which are used as the initialization parameters for the Clustering Layer.

Phase 1-4: Fine-tuning. Based on Eq. (8) and using the training data, our network model is fine-tuned.

Phase 2: Testing. By inputting the testing data into the fine-tuned DataDecoder in Figure 1(c), the soft probability p_{ij} of any data can be obtained according to Eqs. (3) and (4), and the affiliation is thus determined following the majority principle.

4. Experimental studies

4.1. Datasets

Six datasets were involved in the experiment. They are briefly introduced as follows, and their primary attributes are additionally listed in Table 1.

Table 1. Primary attributes of involved 6 datasets.

Dataset Name	#Points	#Classes	#Size	#Input Dimension
MNIST	70000	10	28*28	784
MNIST2	70000	10	28*28	784
USPS	9298	10	16*16	256
FMNIST	70000	10	28*28	784
OMNIST(A)	58850	11	28*28	784
OMNIST(C)	23660	11	28*28	784
OMNIST(S)	25221	11	28*28	784
COIL20	1440	20	32*32	1024

MNIST (Lecun & Bottou, 1998)**:** There are 70,000 handwritten digit pictures in this dataset, and each picture has $28 \times 28 = 784$ pixels. In our experiments, each picture was reshaped into a 784-dimensional vector as input to the autoencoders.

MNIST2 (Wang et al., 2016)**:** Following the work in (Song et al., 2017), the MNIST2 dataset was derived from MNIST and polluted by noise. It has the same attributes as MNIST.

USPS: The USPS dataset has in total 9,298 handwritten digit pictures, and each picture has $16 \times 16 = 256$ pixels. Likewise, each picture was reshaped into a 256-dimensional vector as input to the autoencoders.

FashionMNIST (Xiao et al., 2017)**:** This dataset contains 10 types of daily clothes images, and it has the same attributes as MNIST.

OrganMNIST (Bilic et al., 2019): This medical dataset iscomposed of images of 11 human organs. Each image has three derivatives: the axial, coronal, and sagittal slices, and each slice has $28 \times 28 = 784$ pixels. The images

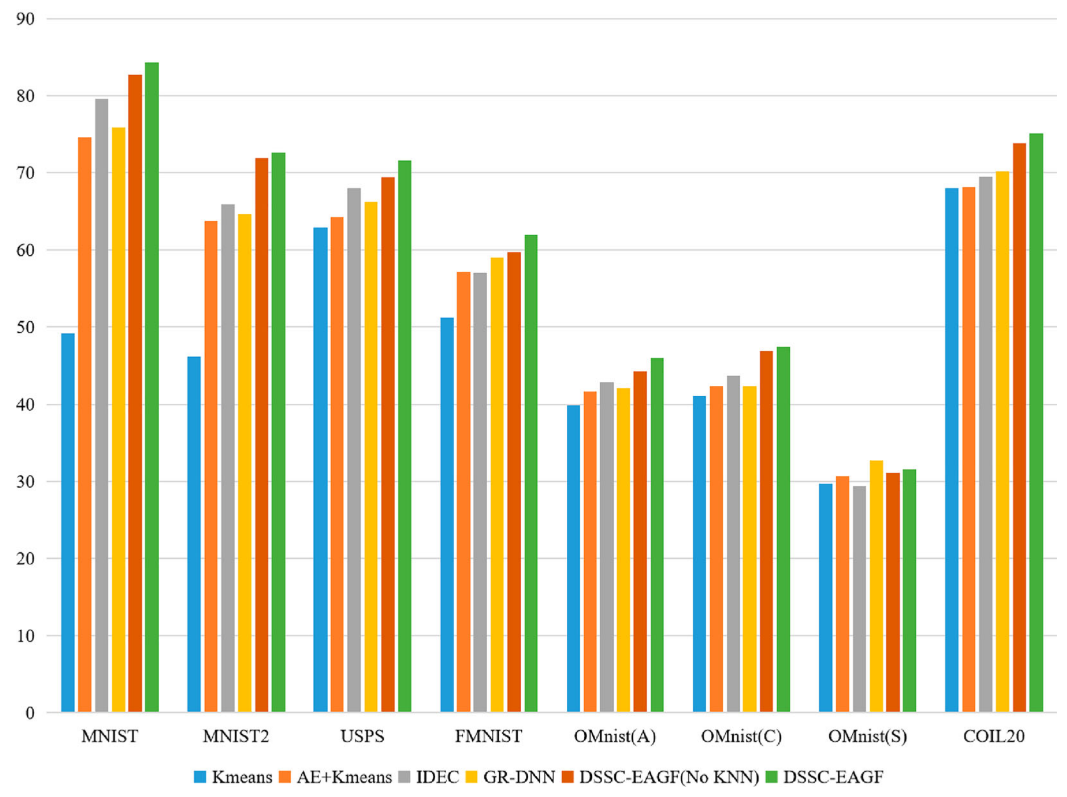

Figure 4. Clustering results (%): NMI.

were reshaped a 784-dimensional vector as input to the autoencoder**s.**

COIL20 (Nene, 1996)**:** The COIL20 dataset contains 1,440 gray images, which are obtained by gradually rotating each raw image (20 in total) 360 degrees in the step of 15 degrees. Each image has $32 \times 32 = 1024$ pixels.

4.2. *Compared methods*

In the experiments, several deep clustering methods similar to our method were taken for performance comparison.

K-means (Macqueen, 1967)**.** *K*-means is a classic clustering algorithm usually used as a baseline for performance comparisons.

AE + Kmeans. This method contains two stages: autoencoder (AE)-leveraged feature extraction and *K*-means clustering on the AE features.

GR-DNN (Yang et al., 2017). Similar to the AE + Kmeans approach, this method first references the adjacent graph information to train the deep network and then runs K-means on the extracted deep features.

IDEC (Guo et al., 2019). As an improved version of DEC (Xie et al., 2016), this method adds the reconstruction loss term to the original loss function of DEC. Accordingly, the network structure contains the data recovery process.

DSSC-EAGF. This is the complete version of our proposed method.

DSSC-EAGF-noKNN. To fully demonstrate the effectiveness of our proposed DSSC-EAGF method, ablation experiments were conducted by removing the KNN self-supervised mechanism, and this derivative is denoted as DSSC-EAGF-noKNN.

4.3. *Parameters setting*

Our network inherently involves two parts: DataAE and GraphAE, and they are both set as fully-connected multilayer perceptron (MLP). For MNIST-related datasets, the dimensionalities of the four layers of our DataEncoder are respectively set to 500, 500, 2000, and K_2. DataDecoder is the inverse of DataEncoder, so the dimensionalities of the four layers of MLP are respectively set to K_2, 2000, 500, and 500. The dimensionalities of the four layers of GraphDecoder are set to 1000, 500, 250, and K_2, respectively. Taking MNIST as an example, the optimization experiments of the dimension K_2 are carried out in (D'Andrade, 1978; Du et al., 2005; Jie et al., 2018; Macqueen, 1967; Mahardhika et al., 2016; Yang et al., 2016; Yang et al., 2017; Zhang et al., 2018), [50]. The results are shown in the Figure 5. It should be noted that the Abscissa in Figure 5 is not equal. In the experimental results, it is not difficult to find that when the data dimension of the deep feature is too small, the clustering effect is not ideal. Because the final feature contains too little information of the original data. After the feature dimension reaches

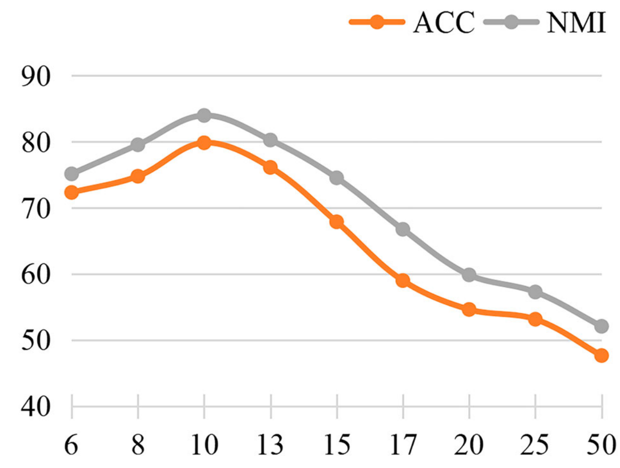

Figure 5. Clustering results (%) of different K_2 values: NMI, ARI.

Table 2. Clustering results (%): NMI, ARI.

		Kmeans	AE+ Kmeans	IDEC	GR-DNN	DSSC-EAGF (No KNN)	DSSC-EAGF
MNIST	NMI	49.1	74.6	79.6	75.9	82.7	**84.3(±1.34)**
	ARI	36.1	70.8	71.1	71.1	73.8	**78.1(±1.11)**
MNIST2	NMI	46.1	63.7	65.9	64.6	71.9	**72.6(±0.83)**
	ARI	33.5	54.8	45.6	51.2	57.6	**59.1(±0.42)**
USPS	NMI	62.9	64.2	68.0	66.2	69.4	**71.6(±0.55)**
	ARI	55.2	54.4	56.0	57.0	57.5	**61.2(±0.37)**
FMNIST	NMI	51.2	57.1	57.0	59.0	59.7	**61.9(±0.39)**
	ARI	34.7	40.7	38.6	42.4	44.1	**46.8(±0.15)**
OMnist(A)	NMI	39.9	41.7	42.9	42.1	44.3	**45.9(±0.48)**
	ARI	21.5	26.6	26.1	28.2	20.7	**32.5(±0.42)**
OMnist(C)	NMI	41.1	42.4	43.7	42.4	46.8	**47.4(±0.24)**
	ARI	21.2	21.7	16.9	20.0	25.7	**26.4(±0.16)**
OMnist(S)	NMI	29.7	307	29.4	**32.7**	31.1	31.6(±0.40)
	ARI	14.1	13.9	12.3	**16.0**	13.8	13.9(±0.34)
COIL20	NMI	68.0	68.1	69.5	70.2	73.8	**75.1(±0.73)**
	ARI	40.9	45.6	46.6	45.2	56.9	**57.3(±0.52)**

a peak, we can clearly see that the clustering results start to deteriorate as the feature dimension increases. At the same time, the dimension of deep features also affects the network scale of ClusteringLayer, and increasing the dimension will also increase the complexity of parameter update. Therefore, we set the dimension of deep features in MNIST to 10.

The nonlinear function ReLU (Mukhametkha et al., 2018) is used as the activation function. The pre-training procedure of our network is introduced in Section 3.5. After pre-training, the hyperparameter λ that controls the weight of clustering loss was set to 0.1 (Guo et al., 2019), and the batch size was set to 1000 on all datasets. To verify the setting of the superparameter λ in (Guo et al., 2019), an optimization experiment was conducted for λ. In the experiment, the alternative value of λ was {0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1.0}, and the experimental results will be analyzed in Section 4.4. The optimizer Adam (Kingma & Ba, 2014) with an initial learning speed $lr = 0.001$ was applied to the MNIST-related datasets, and β_1 and β_2 were respectively set to the default values of 0.9 and 0.999. SGD was applied to the USPS dataset, and lr and β were respectively set to 0.1 and 0.99. The convergence threshold δ was set to 0.001. In the KNN self-supervised module, the value of K_3 was set to 7, and it was determined by the grid search within $\{3, 5, 7, 9, 11\}$. Meanwhile, the coefficient γ of the self-supervision loss was set to 0.1, and it was determined by the grid search within {0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1.0}.

Evaluation indicators: two well-established performance indicators, i.e. NMI and ARI, were adopted to evaluate the clustering effect of the methods.

In addition, all the methods were implemented with Python and Pytorch.

Adjacent Graph Features: As mentioned in Section 3.1, the values of N_a and K_1 have an important impact on obtaining the global characteristic. Throughout our empirical studies, it was found that keeping N_a close to the dimensionality of the original training data helps to maintain the data characteristics during the whole encoding and decoding process. Therefore, on MNIST ($d = 784$), N_a was set to 1000. In this case, 1000 anchor points would be evenly distributed in the original data space, which facilitates keeping the ground truth of the original adjacent graph.

After N_a was determined, the setting of K_1 is introduced as follows by taking MNIST as an example. When Na was set to 1000, K_1 points that can effectively maintain the global characteristic need to be selected. It is required that most of these points belong to the same cluster and have a similar distribution in the global space. If K_1 is too large, the adjacency graph overemphasizes the global structure and introduces too many non-clustered anchor points, leading to a great sparseness to the adjacent graph and losing the effective affinity. In contrast, if K_1 is too small, the constructed adjacency graph overemphasizes the local structure. In this case, the selected anchor points are very similar to the original data, and the global structure is weakened, which degrades the effectiveness of the adjacency graph. Therefore, in the experiment, K_1 was set to 30 on MNIST and 15 on USPS, respectively.

4.4. Experimental results and analyses

Table 2 shows the clustering results of all comparison methods on the six datasets. It can be seen in Figure 4 that the algorithms based on deep learning, including AE + Kmeans, DEC, IDEC, GR-DNN, and DSSC-EAGF, perform significantly better than the traditional clustering algorithm Kmeans. This indicates that deep learning has a very potential for unsupervised clustering.

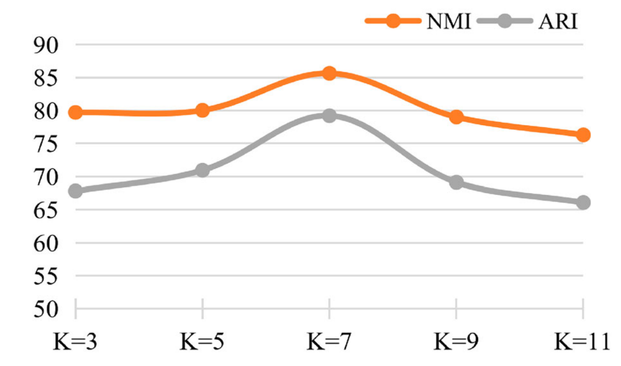

Figure 6. Clustering results (%) of different K_3 values: NMI, ARI.

Table 3. Clustering results (%) under different K_3 values: NMI, ARI.

	K = 3	K = 5	K = 7	K = 9	K = 11
NMI	79.71	80.03	**85.64**	79.02	76.30
ARI	67.82	70.94	**79.21**	69.13	66.09

The comparison of the clustering results of AE + Kmeans and DEC indicates that IDEC performs better than AE + Kmeans. Such improvement is certainly attributed to the positive influence of the clustering module. The clustering module parameterizes the centres of all clusters, and such cluster centres are more representative than those of direct selection.

Then, AE + Kmeans is compared with GR-DNN. Both use the autoencoder to reduce data dimensions and combine the traditional clustering method for model optimization, but the latter embeds the adjacency graph information. The adjacency graph represents certain structural information of the global distribution of the data. Leveraging the encoding features of the adjacency graph in the low-dimensional space for self-supervised learning effectively improves the network performance. Through the experiments on multiple datasets, it is found that GR-DNN achieves better clustering results better AE + Kmeans, which confirms our analyses.

Besides, an ablation experiment i.e. DSSC-EAGF-NoKNN, was conducted to determine the effectiveness of the KNN self-supervised module proposed in this paper. The comparison of the results between DSSC-EAGF-NoKNN and DSSC-EAGF is shown in Table 2. It can be seen that the latter performs better than the former, showing the effectiveness of the KNN self-supervised module. Table 3 further shows that when the value of K_3 is too large or too small, the result is affected, and both algorithms obtain poor performance. When K_3 is very small, the module has little influence on the network performance, compared with DSSC-EAGF-NoKNN (see the case: $K_3 = 1$). When K is too large, some data points that do not belong to the same type may be involved in the calculation, thus inducing wrong supervision information.

The clustering loss and self-supervised loss were set to 0.1 (by the grid search in {0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0} shown in Figure 8(a)). The experimental results show that when the parameter λ of clustering loss is set to 0.01, 0.02, and 0.05, the clustering loss has little effect on backpropagation and a limited influence on network parameter update. As a result, the final network training does not achieve good results. Meanwhile, when λ is set to 0.2, 0.5, and 1.0, the updating of network parameters focuses on the fitting clustering layer, which makes the autoencoder gradually fail to recover data. The model overfits the clustering layer, resulting in poor clustering results. Figure 8(b) shows that, as for the parameter γ of self-supervised loss, its influence on the clustering effect reaches the maximum when it is set to 0.1. This is because when the γ value is small, the self-supervised loss has little influence on the network, which is similar to the case when the K_3 value is small. However, as the γ value increases gradually, the effect of clustering loss on the parameter setting of the network and the location of the clustering centre (namely, the parameter in the Clustering Layer) is gradually offset. Finally, the network overfits in another direction, which is different from the case when λ is too large.

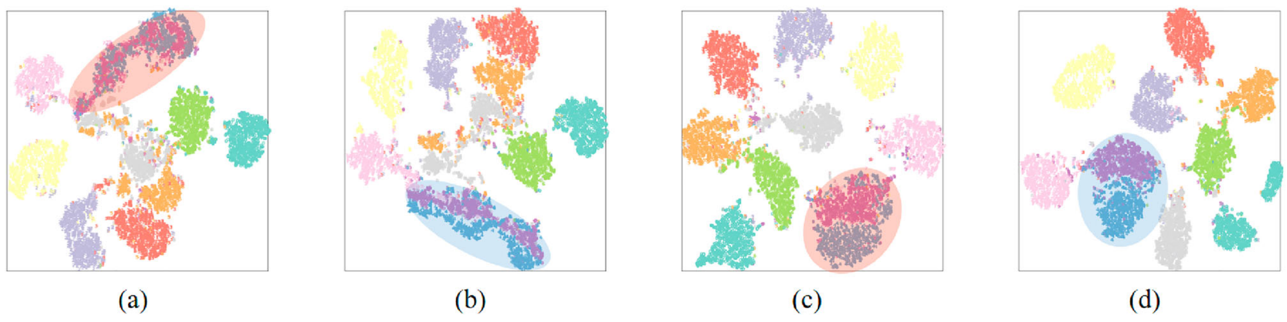

Figure 7. The two-dimensional Visualization of IDEC and DSSC-EAGF on the MNIST dataset. (a) and (b) are the visualization of IDEC in the training set and testing set of MNIST. (c) and (d) are the visualization of DSSC-EAGF in the training set and testing set of MNIST.

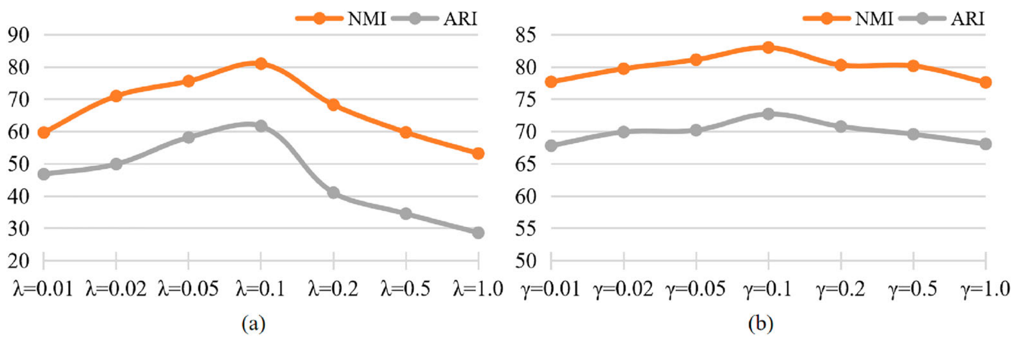

Figure 8. Clustering results (%) under different λ (a) and γ (b): NMI, ARI.

To visualize the training results, as shown in Figure 7, the t-SNE (Wu et al., 2017) is employed to display the 10,000 data instances extracted from the MNIST training and testing sets. Figure 7(a) and (b) illustrate the visualization of IDEC, and Figure 7(c) and (d) illustrate the visualization of DSSC-EAGF. The data points under the pink shadow in Figure 7(a) and Figure 7(c) represent numbers 4 and 9 in the training set. They belong to the same part of data but are subject to dimension reduction by different encoders. Similarly, the data points shade in green in Figure 7(b) and Figure 7(d) are numbers 4 and 9 in the testing set.

It can be seen that DSSC-EAGF well maintains the shape of each cluster. On the contrary, the comparison of the pink and green shadows in Figure 7 indicates that IDEC fails to differentiate 4 and 9 well, and there is even a little overlapping.

All the above results verify that DataAE can retain the internal structure of the data, whereas GraphAE can mine the global structure of the original data space. Besides, the KNN self-supervised loss can help to properly optimize the networks in the low-dimensional encoding feature space.

5. Conclusions

This paper proposes the DSSC-EAGF algorithm to facilitate the clustering task. DSSC-EAGF maintains the global structure and learns embedded features suitable for clustering. Also, it manipulates the feature space to divide the data distribution by optimizing the clustering loss and self-supervised loss based on KL divergence. Because our algorithm is an improved version of the IDEC algorithm, The experimental results on various datasets show that our algorithm achieves a better clustering effect than IDEC. Meanwhile, the experiment proves that the maintenance of the global structure is beneficial to improve the clustering performance, and the KNN self-supervised module also shows its effect.

At the same time, we also found our own shortcomings. In the experiments, we all use the processed grayscale images, which is due to the limitations of the fully connected network in processing colour images. If the fully connected network is replaced by a convolutional network, the overall performance will inevitably be improved. However, whether the convolutional network can effectively cooperate with the adjacency graph features to improve the clustering effect still needs to be experimentally verified.

To improve our algorithm, the following work will be done in future work: 1) Adding some prior knowledge to the framework; 2) Replacing the fully connected network with a convolutional network for image processing.

Data availability statement

The data that support the findings of this study are available from the corresponding author, upon reasonable request.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported in part by the Science and Technology Demonstration Project of Social Development of Jiangsu Province, China under Grant [grant no BE2019631].

References

Belkin, M., & Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, *15*(6), 1373–1396. doi:10.1162/089976603321780317

Bickel, S., & Scheffer, T. (2004). *Multi-view clustering. Data Mining, 2004*. ICDM'04. Fourth IEEE International Conference on. IEEE.

Bilic, P., Christ, P. F., Vorontsov, E., Chlebus, G., Chen, H., Dou, Q., Fu, C. W., Han, X., Heng, P. A., Hesser, J., & Kadoury, S. (2019). The Liver Tumor Segmentation Benchmark (LiTS).

D'Andrade, R. G. (1978). *U-statistic hierarchical clustering*. Psychometrika.

Du, J. X., Huang, D. S., Wang, X. F., & Gu, X. (2005). *Neural network-based shape recognition using generalized differential evolution training algorithm*. IEEE International Joint Conference on Neural Networks. IEEE.

Guo, X., Gao, L., Liu, X., & Yin, J. (2019). *Improved Deep Embedded Clustering with Local Structure Preservation*. International Joint Conference on Artificial Intelligence.

Hadsell, R., Chopra, S., & Le Cun, Y. (2007). *Dimensionality Reduction by Learning an Invariant Mapping*. International Conference on Thermoelectrics.

Han, F., Ling, Q.-H., & Huang, D.-S. (2010). An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks. *Neural Computing & Applications*, *19*(2), 255–261. doi:10.1007/s00521-009-0274-y

Han, F., Ling, Q. H., & Huang, D. S. (2008). Modified constrained learning algorithms incorporating additional functional constraints into neural networks. *Information Sciences: An International Journal*, *178*(3). doi:10.1016/j.ins.2007.09.008

Hao, Y., Han, C., Shao, G., & Guo, T. (2013). *Generalized Graph Regularized Non-negative Matrix Factorization for Data Representation*. Springer Berlin Heidelberg.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. doi:10.1126/science.1127647

Huang, D.-S. (2011). Radial Basis Probabilistic Neural Networks: Model and Application. *International Journal of Pattern Recognition and Artificial Intelligence*, *13*(7). doi:10.1142/S0218001499000604

Jie, L., Junyu, X., Guangquan, Z., & Xiangfeng, L. (2018). Structural property-aware multilayer network embedding for latent factor analysis. *Pattern Recognition: The Journal of the Pattern Recognition Society*, *76*. doi:10.1016/j.patcog.2017.11.004

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv e-prints.

Lecun, Y., & Bottou, L. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. doi:10.1109/5.726791

Li, Y., Nie, F., Huang, H., & Huang, J. (2015). *Large-Scale Multi-View Spectral Clustering via Bipartite Graph*. AAAI Conference on Artificial Intelligence.

Luxburg, U. V. (2004). A Tutorial on Spectral Clustering. *Statistics and Computing*, *17*(4), 395–416. doi:10.1007/s11222-007-9033-z

Macqueen, J. B. (1967). *Some methods for classification and analysis of multivariate observations*. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967. University of California Press.

Mahardhika, P., Jie, L., Sreenatha, A., Edwin, L., & Chee-Peng, L. (2016). An incremental meta-cognitive-based scaffolding fuzzy neural network. *Neurocomputing*, *171*(JANa1), 89–105. doi:10.1016/j.neucom.2015.06.022

Mukhametkhan, M., Krestinskaya, O., & James, A. (2018). Analysis of Multilayer Perceptron with Rectifier Linear Unit Activation Function. 245–49. doi:10.1109/CoCoNet.2018.8476902

Nene, S. A. (1996). Columbia Object Image Library(COIL-20). *Technical Report*, *5*.

Nigam, K., & Ghani, R. (2002). Analyzing the Effectiveness and Applicability Of Co-Training.

Pei, S., & Huang, D.-S. (2006). Cooperative Competition Clustering for Gene Selection. *Journal of Cluster Science*, *17*(4), 637–651. doi:10.1007/s10876-006-0077-6

Song, Y., Zhang, G., Lu, J., & Lu, H. (2017). *A fuzzy kernel c-means clustering model for handling concept drift in regression*. IEEE International Conference on Fuzzy Systems. IEEE.

Wang, C., Lu, J., & Zhang, G. (2007). *A Constrained Clustering Approach to Duplicate Detection Among Relational Data*. Pacific-asia Conference on Advances in Knowledge Discovery & Data Mining. Springer-Verlag.

Wang, W., Arora, R., Livescu, K., & Bilmes, J. (2016). On Deep Multi-View Representation Learning: Objectives and Optimization.

Wang, X., & Huang, D. S. (2009). A Novel Density-Based Clustering Framework by Using Level Set Method. *IEEE Transactions on Knowledge and Data Engineering*, *21*(11), 1515–1531. doi:10.1109/TKDE.2009.21

Wu, J., Wang, J., Xiao, H., & Ling, J. (2017). Visualization of High Dimensional Turbulence Simulation Data using t-SNE. doi:10.2514/6.2017-1770

Xianpeng, L., Di, W., & De-Shuang, H. (2019). Image Co-Segmentation via Locally Biased Discriminative Clustering. *IEEE Transactions on Knowledge and Data Engineering*, *31*(11), 2228–2233. doi:10.1109/TKDE.2019.2911942

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

Xie, J., Girshick, R., & Farhadi, A. (2016). *Unsupervised Deep Embedding for Clustering Analysis*. International Conference on Machine Learning.

Yang, J., Parikh, D., & Batra, D. (2016). Joint Unsupervised Learning of Deep Representations and Image Clusters. *IEEE Conference on Computer Vision & Pattern Recognition*, 5147–5156. doi:10.1109/CVPR.2016.556

Yang, S., Li, L., Wang, S., Zhang, W., & Huang, Q. (2017). *A Graph Regularized Deep Neural Network for Unsupervised Image Representation Learning*. IEEE Conference on Computer Vision and Pattern Recognition.

Yu, H., Lu, J., Xu, J., & Zhang, G. (2019). *A Hybrid Incremental Regression Neural Network for Uncertain Data Streams*. 2019 International Joint Conference on Neural Networks (IJCNN).

Yu, H., Lu, J., Zhang, G., & Wu, D. (2018). *A Dual Neural Network Based On Confidence Intervals For Fuzzy Random Regression Problems*. IEEE International Conference on Fuzzy Systems.

Zhang, J., Li, C. G., You, C., Qi, X., Zhang, H., Guo, J., & Lin, Z. (2019). *Self-Supervised Convolutional Subspace Clustering Network*. IEEE.

Zhang Yi, Lu Jie, Liu Feng, Liu Qian, Porter Alan, Chen Hongshu, Zhang Guangquan (2018). Does deep learning help topic extraction? A kernel k-means clustering method with word embedding. *Journal of Informetrics*, *12*(4), 1099–1117. doi:10.1016/j.joi.2018.09.004

Zhao, Z. Q., Huang, D. S., & Sun, B. Y. (2004). Human face recognition based on multi-features using neural networks committee. *Pattern Recognition Letters*, *25*(12), 1351–1358. doi:10.1016/j.patrec.2004.05.008