Applied Intelligence
https://doi.org/10.1007/s10489-022-03895-5

EDCWRN: efficient deep clustering with the weight of representations and the help of neighbors

Amin Golzari Oskouei[1] · Mohammad Ali Balafar[1] · Cina Motamed[2]

Accepted: 13 June 2022

Abstract
In existing deep clustering methods, it is assumed that all generated representations are equally important during the clustering procedure. However, if the model can't learn proper cluster-oriented representations, all generated representations may not be suitable for clustering. In this case, some important representations need to be more effective than the other representations in forming optimal clusters. The existing deep clustering methods do not support this idea. Also, in most methods, Kullback–Leibler Divergence (KLD) loss function is used. KLD does not preserve global data structure. In this paper, an efficient joint deep clustering framework, termed EDCWRN, is introduced to learn representations and cluster labels, simultaneously. To overcome the mentioned problems, in EDCWRN, an automatic local representation weighting strategy is applied to weight the representations of each cluster properly. Moreover, the samples and their neighbors are involved in the learning representations procedure to generate better representation. Also, a new efficient formulation for cluster assignments is proposed. Using this formulation, global and local data structure is preserved simultaneously. Experiments show that the proposed model is more efficient than other state-of-the-art methods. The implementation-source code- of EDCWRN is made publicly available at https://github.com/Amin-Golzari-Oskouei/EDCWRN.

Keywords Deep learning · Clustering · Deep clustering · Local feature weighting

1 Introduction

Clustering is an important data analysis technique used to get an intuition about the data structure [1–5]. Clustering is grouping a set of samples into several partitions (clusters) [6, 7]. The goal of clustering is to decrease inter-cluster similarity and increase intra-cluster similarity [6, 8–10]. Various classic clustering algorithms have been introduced, in recent years. The fuzzy c-means [12] and k-means [12] clustering algorithms are the most popular clustering algorithms among the classic clustering methods. Some other extensions of these algorithms have also been introduced in [3, 13, 14].

Traditional clustering methods (such as fuzzy c-means [12] and k-means [12]) usually do not form optimal clusters for high dimension datasets owing to the inefficiency related to the similarity criteria used in these methods. Such methods also have high computational complexity for large-scale datasets. Therefore, dimensionality reduction techniques (such as principal component analysis (PCA) [15]) are used to map raw data into a low-dimensional feature space. However, a highly complex latent structure of data still challenges the usefulness of current clustering methods [16].

Thanks to deep learning development, deep neural networks (DNNs) can be implemented to transform data into more clustering-friendly representations as they have an inherent property of highly non-linear transformation [17–19]. Clustering methods that utilize deep learning approaches are called deep clustering [20]. Most deep clustering methods consist of two phases. The first phase is learning new representations of raw data using deep learning models, and the second phase is the clustering of this data in the new embedded space. In the second phase, both classical clustering

✉ Mohammad Ali Balafar
balafarila@tabrizu.ac.ir

Amin Golzari Oskouei
a.golzari@tabrizu.ac.ir

Cina Motamed
motamed@free.fr

[1] Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz 5166616471, Iran

[2] Department of Computer Science, University of Orleans, Orléans, France

Published online: 05 July 2022

techniques and deep learning models can be used to identify clusters. Deep clustering methods have recently attracted researchers thanks to some key advantages that they have [21]; 1) generated representations of data are in low dimensional space and have better information for clustering (clustering-friendly representations). As a result, clustering in this new embedded space is much faster and better than in the raw data space, and 2) for clustering complex data (such as images or text), using new representations significantly improves clustering performance.

Problem statement In existing deep clustering methods, it is assumed that all generated representations are equally important during the clustering procedure. However, if the model can't learn proper representations, all generated representations may not be suitable for clustering. Therefore, some generated representations are irrelevant to the target problem or less important than others. However, most deep clustering methods do not consider the importance of the generated representations, and as a result, in some applications, their performances fall significantly.

To demonstrate the importance of the generated representations in each cluster, we illustrate the representations generated by the IDEC [22] method on the *MNIST* dataset in different 2D subspaces. Figure 1 shows the result of this visualization. Note that for simplicity, this visualization is only illustrated on three classes and four generated features. From this figure, it is understood that Cluster2 is formed chiefly based on Feature1 and Feature4 (see Fig. 1c), and Cluster1 is mainly formed based on Feature3 and Feature4 (see Fig. 1b). This means that the first and the fourth features in Cluster2 are more important than the second and the third features (see Fig. 1d). Similarly, the third and the fourth features in Cluster1 are more important than the first and the second features (see Fig. 1a).

In addition, most deep clustering methods, such as [22–24], use the Kullback–Leibler Divergence (KLD) loss function to train the model in the clustering step, and the Student's t-distribution is used to predict the probability of assigning a sample to a cluster (soft assignment). KLD does not preserve global data structure, which means only within-cluster distances are significant, while between-clusters similarities are not guaranteed. Therefore, it is generally agreed that clustering by the KLD loss function is not a very good idea. Also, because the Student's t-distribution is not scaled and should be normalized, using the Student's t-distribution increases model training time.

To better understand the mentioned problem, we use a one-dimensional dataset with 100 samples (0 to 99) and two clusters (centers 25 and 75). By illustrating the KLD loss function with the Student's t-distribution (see Fig. 2), it is observed that for samples close to the centers (such as samples 20 to 30 and 70 to 80), the loss function behaves correctly. By moving away from the center of the clusters, the loss increases.

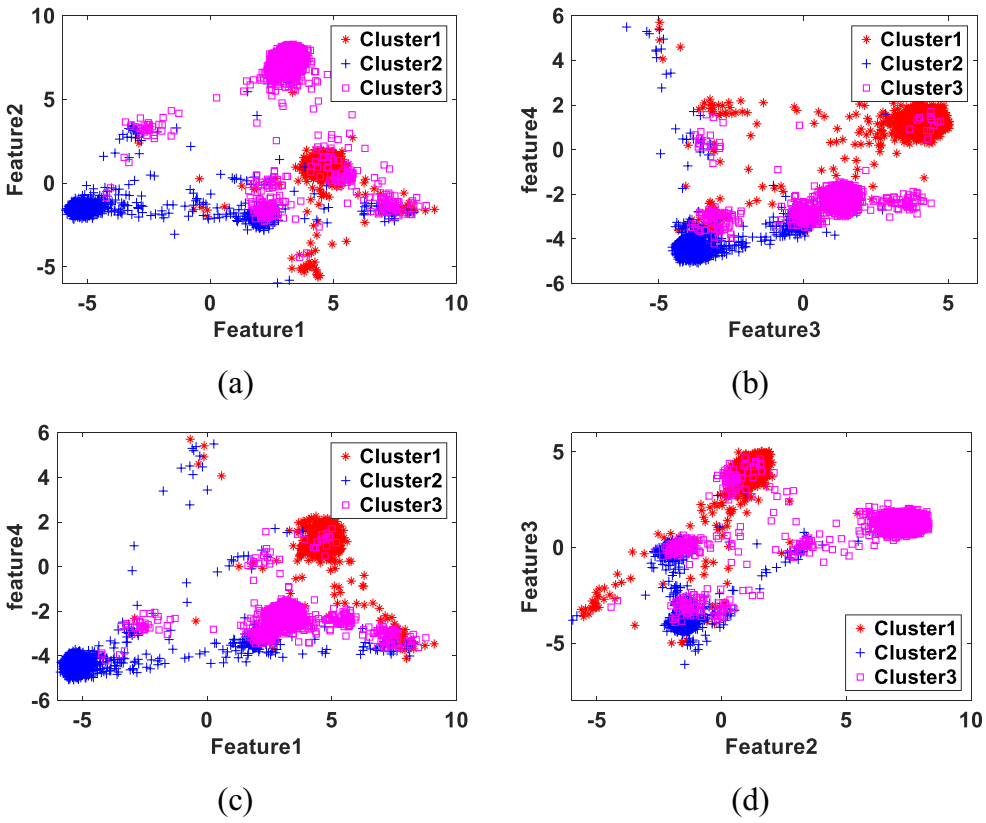

Fig. 1 Visualization of the representations generated by the IDEC [22] method on the *MNIST* dataset in different 2D subspaces

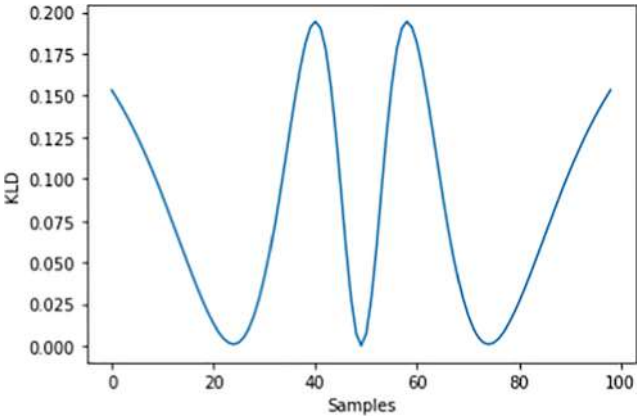

Fig. 2 KLD loss function behavior

However, for samples far from the centers (such as samples 40 to 60), the loss function does not behave correctly. By moving away from the centers of the clusters, the loss either increases or decreases. While we expected that by moving away from the centers of the clusters, the loss increases and reaches its maximum value for the 50th sample. Therefore, KLD does not preserve global data structure, meaning that only within-cluster distances are meaningful, while between clusters similarities are not guaranteed.

Method To tackle the problems of deep clustering methods (i.e., equal importance to generated representations, KLD loss function, and Student's t-distribution problems), in the proposed deep clustering framework, we use an automatic local representation weighting strategy to weight the representations of each cluster properly. Representation weighting is conducted locally so that depending on their importance in the clusters, the representations would have different weights, increasing the quality of clustering. Moreover, we utilize an efficient deep model to learn representations and cluster labels jointly. In the proposed deep clustering framework, the samples and their neighbors are used in the learning representations procedure to generate cluster-oriented meaningful representation. The advantage of utilizing data neighbors is that for similar samples (in the same cluster), the representations generated in the embedded space are close to each other. Also, in the proposed model, instead of using the KLD loss function, we use cross-entropy. Using the cross-entropy loss function, global and local data structure is preserved simultaneously, meaning that intra-cluster and inter-cluster distances in the generation of new representations are considered. In the proposed method, the formulation for cluster assignments (cluster labels) is new and different from the Student's t-distribution, which has significantly improved the results.

The graphical abstract of the proposed method is shown in Fig. 3. As shown in this figure, the proposed method consists of two steps: pre-training and clustering. In the pre-training step, the autoencoder model is trained, then this model is used in the clustering step. In the clustering step, the sample and its k nearest neighbors are used to train the model. The main aim is to learn similar representations for samples belonging to the same cluster. The output of the encoder module is fed to the clustering layer to predict clusters. In this layer, the local features weighting technique is used to identify more important features. This weighting helps to form optimal clusters.

Results Extensive experiments conducted on six standard datasets indicate all used techniques somehow positively affect the final result. The advantage of these techniques is that the algorithm achieves optimal clusters by performing the feature weighting along with the proposed formulation and neighboring concurrently. Feature weighting helps select more important features for clustering, the neighborhood technique helps to learn similar representations for samples belonging to the same cluster, and the cross-entropy loss function with the proposed distribution helps preserve the inter-cluster distance (global distance). Representation weighting, proposed formulation, and applied neighboring techniques have the most positive effect on the final results, respectively. Comparing the results to the existing state-of-the-art deep clustering methods shows significant performance.

Contribution This research makes the contributions as follows:

1) Representation weighting is conducted locally depending on their importance in the clusters; the representations in each cluster have different weights, hence increasing the quality of clustering;
2) An efficient deep clustering framework is designed to learn representations and cluster labels jointly. Also, the samples and their neighbors are used in the learning representations procure to generate cluster-oriented meaningful representation;
3) Global and local data structure is preserved, meaning that intra-cluster and inter-cluster distances in the generation of new representations are considered;
4) The formulation for cluster assignments (cluster labels) is new and different from the other methods, which has significantly improved the results.

The remainder of the paper is organized as follows: In Section 2, some existing deep clustering methods are reviewed; in Section 3, a brief overview of the IDEC [22] method is presented, which is the basis of this work; in Section 4, the proposed deep clustering model (EDCWRN) is described in detail; Section 5 presents the experiments; finally, Section 6 makes conclusion and ideas for future work.

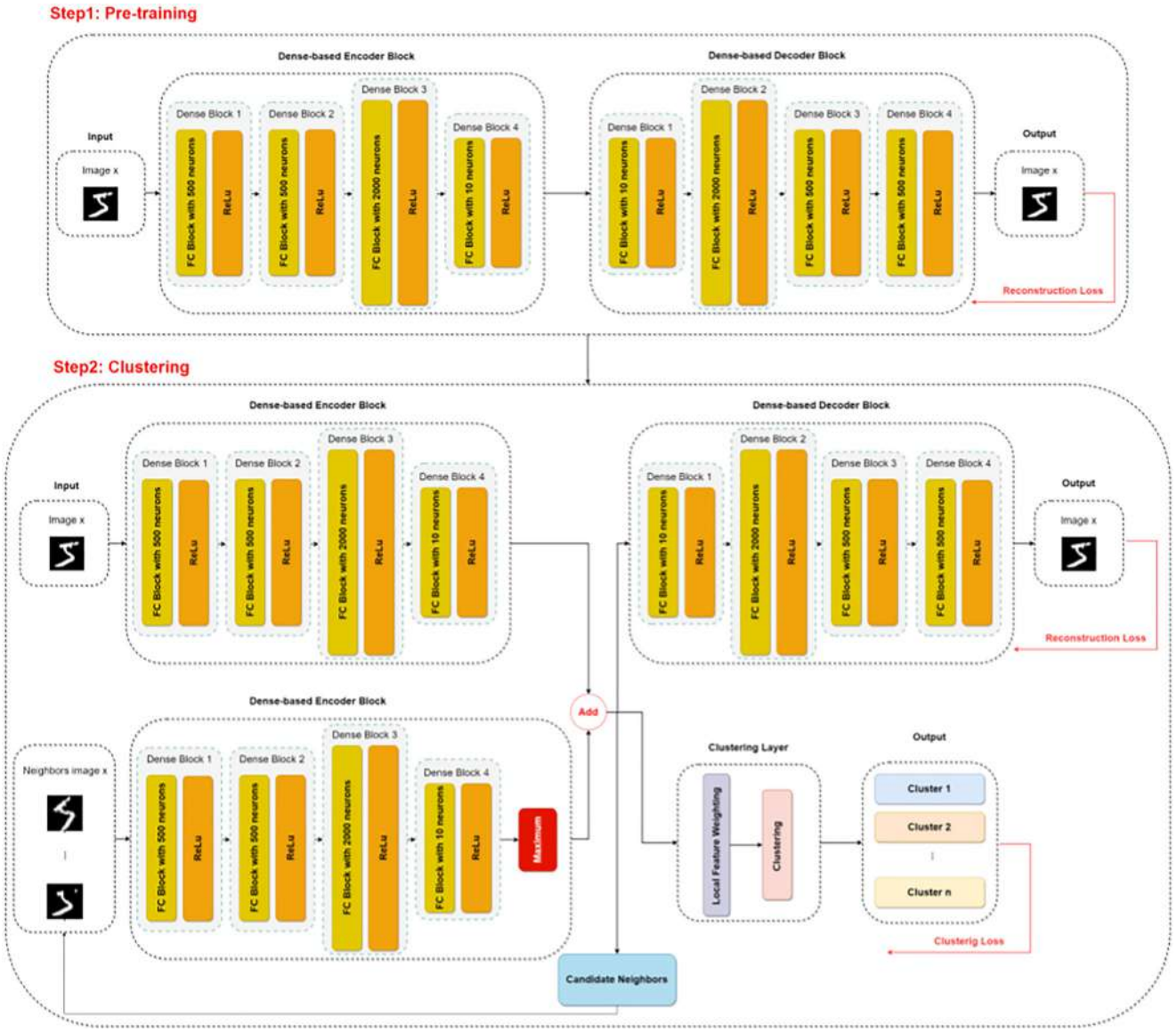

Fig. 3 Graphical abstract of the proposed method

2 Related work

There are two approaches to clustering data based on deep learning. The first approach can use extracted features from a deep pre-trained network. In this approach, the extracted features are usually not appropriate for unsupervised tasks. In fact, the usage of an already-trained deep network (by labeled data) for some unsupervised purposes lacks knowledge of features required for partitioning unknown data [21]. The second approach has a clustering method embedded in a deep learning model. This connection allows the deep learning model to learn clustering-friendly representations.

Some models are trained by an auxiliary target distribution. Using auxiliary target distribution during network training improves clustering accuracy. Deep Embedded Clustering (DEC) [27] is the first algorithm to use auxiliary target distribution. In this algorithm, a fully connected deep autoencoder model is trained by the MSE (Mean Square Error) loss function. At the end of the training, the network decoder is removed, and a new layer called the clustering layer is added to the top of the network. The new network is trained with the KLD loss function, which is calculated between predicted cluster assignment probability and auxiliary target distribution. For designing the auxiliary distribution, a function of the current model assignment distribution and the frequency per cluster is used by the authors. In DEC, the Student's t-distribution is used to predict the probability of assigning a sample to a cluster (soft assignment).

Some other extensions of DEC methods have been introduced. DEC with data augmentation (DEC-DA) [29] is a new DEC-based method that uses data augmentation during the training autoencoder model. Discriminatively Boosted

Clustering (DBC) [27] is another clustering method with the same framework as DEC, except that it uses CNN layers instead of fully connected layers. Because DBC uses CNN layers, it works much better than DEC for clustering image datasets. IDEC [22] is another DEC-based approach that tries to preserve the local data structure. This algorithm uses MSE and KLD loss functions simultaneously to train the proposed network. As a result, it generates better representations for clustering. We have chosen this algorithm as the primary base for comparing our proposed framework. In Section 3, the IDEC method, i.e., the basis of the suggested method (EDCWRN), is introduced.

Deep Multi-Manifold Clustering (DMC) [31] is a deep learning-based multi-manifold clustering approach that proposes a loss function based on three elements. The first element is the clustering loss which makes the data representations cluster friendly. The second term in the loss is the autoencoder loss which helps to obtain new data representations, and finally, a locally preserving loss which makes the representations useful and meaningful.

Existing DEC-based methods use pseudo-labels (auxiliary target distribution) to train the network in the clustering phase. However, the estimation of these pseudo-labels may not be correct in the early epochs of model learning, and the network may not be appropriately optimized. To address this issue, an improved deep convolutional embedded clustering algorithm using reliable samples (IDCEC) was proposed in [29]. In the clustering phase of IDCEC, reliable samples were selected and passed to the convolutional neural network for training to get better clustering results.

Dynamic Autoencoder (DynAE) [30] is a deep learning-based clustering approach that overcomes a clustering reconstruction trade-off by gradually and smoothly eliminating the reconstruction objective function in favor of a construction one. Although this method performs better than other similar algorithms, the network training time is longer than other similar methods and suffers from parameter dependence. This group of researchers has proposed other similar methods in [23].

Clustering can be performed by utilizing pairwise constraints [31] as well. The main idea of this paper is that similar pairs should have similar representations in the embedding space while different pairs should have different distant embeddings. The authors have used K-Nearest Neighbor (KNN) to obtain pairwise relations. Maximizing inter-cluster variance and minimizing intra-cluster variance for image datasets have been studied in [21]. In this paper, for each image, one related image (positive sample) and one non-related (negative sample) image are selected. The goal is to make the representation of the input image close enough to the positive sample while it is well separated from the negative one. A positive sample is obtained by the data augmentation technique, and the negative sample is randomly selected from the entire dataset. Therefore, the negative sample may not be determined correctly. Recently, a text clustering method was proposed in [24]. In this method, neighbors preserve cluster locality [24]. In [32], a new constrained document clustering was proposed. In this method, some of the informative data pairs are selected during an iterative process.

In [33], the authors proposed an efficient deep image clustering model using nearest neighbor contrastive, which fuses contrastive learning with neighbor relation mining. During training, contrastive learning and neighbor relation mining are updated alternately, where the former is conducted in the backward pass, while the latter is employed in the forward pass. In this framework, the data augmentation approach is used for generating nearest neighbors manually.

In [34], the authors introduced a new deep framework, called DCV (deep clustering and visualization). The DCV model consists of two non-linear dimensionality reduction (NLDR) transformations: 1) from the input data to the embedded feature space for clustering and 2) from the embedded feature space to the final 2-D space for visualization. The first NLDR transformation is optimized by Clustering Loss, allowing arbitrary corruption of the geometric structure for better clustering. In contrast, the second NLDR transformation is optimized by one Geometry-Preserving Loss to recover the corrupted geometry for better visualization.

In the field of deep clustering, new advanced methods have been proposed that use graph theory and graph convolutional network (GCN) for data clustering. In [35], a robust clustering model based on attention mechanism and graph convolutional network (GCN) was proposed. This model used graph attention network and GCN to learn the feature information of nodes and the topological structure information of graphs, respectively. Then the representation results of the above two learning modules were interactively fused by the interlayer transfer operator. Finally, the model was trained end-to-end using a self-supervised training module to optimize the clustering results. In [36], an end-to-end self-supervised graph convolutional network for multi-view clustering was proposed. This model constructs a new view descriptor for graph-structured data by mapping the raw node content into the complex space via Euler transformation, which not only suppresses outliers but also reveals non-linear patterns embedded in data. Meanwhile, this model uses the clustering labels to guide the learning of the latent representation and coefficient matrix, and the latter, in turn, is used to conduct the subsequent node clustering. In this way, clustering and representation learning are seamlessly connected to achieve better clustering results.

In [37], the authors introduced an efficient model for clustering face images using a residual graph convolutional network, which contains more hidden layers. For each node, the k-Nearest Neighbor (kNN) algorithm was used to construct its sub-graphs. Then the idea of ResNet into GCNs and construct

RGCN was applied to learn the possibility of linkage between two nodes.

TADW-SC [6] and ASC [38] are spectral clustering methods that construct affinity matrices. TADW-SC generates node embedding vectors and builds an affinity matrix by calculating the similarity between each pair of nodes according to the embedding vectors. ASC used biased random walks considering network topology and attributes to calculate the similarity between each pair of nodes for the construction of an affinity matrix. CSADW [39] calculates the topological and attributed similarity between nodes to construct a weighted adjacency matrix and then performs biased random walks according to the matrix.

3 Preliminaries

The improved deep embedded clustering (IDEC) method [22] is a simple and effective clustering method. In this section, the IDEC method is introduced. This algorithm develops a joint architecture for learning representations and cluster labels. Figure 4 shows the IDEC architecture.

As shown in Fig. 4, IDEC is trained simultaneously with two loss functions: a representation loss and a clustering loss. Eq. (1) shows the used loss function in this method. This loss combines a clustering loss ($\mathcal{L}_c$) and a representation loss ($\mathcal{L}_r$). λ is a coefficient, controlling the degree of distorting embedded space.

$$\mathcal{L} = \mathcal{L}_r + \lambda\, \mathcal{L}_c \tag{1}$$

The reconstruction loss function ($\mathcal{L}_r$) in this algorithm is defined in Eq. (2).

$$\mathcal{L}_r(\chi, \eta, \theta) = \sum_{x \in \chi} \delta^I\left(x, g_\eta \circ f_\theta(x)\right) \tag{2}$$

Where χ is a set of input samples, θ and η are respectively the encoder and decoder parameters, $f_\theta(x)$ is the embedding of the data (χ), and $g_\eta \circ f_\theta(x)$ is the reconstructed version of the data obtained from the output layer, δ^I is a dissimilarity measure (e.g., Euclidean, cosine, etc.).

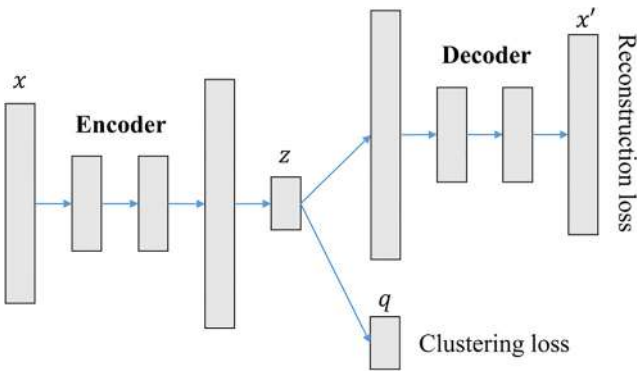

Fig. 4 The architecture of IDEC

The clustering loss function ($\mathcal{L}_c$) in this algorithm (Eq. (3)) is defined as KLD between soft computed labels (Q), which is measured by Student's t-distribution and the target distribution (P) obtained from Q.

$$\mathcal{L}_c = KL\left(P||Q\right) = \sum_i \sum_j p_{ij} \log\left(\frac{q_{ij}}{p_{ij}}\right) \tag{3}$$

In Eq. (3), p_{ij} and q_{ij} are defined in Eqs. (4) and (5), respectively.

$$q_{ij} = \frac{\left(1 + \left\|f_\theta(x_i) - r_j\right\|^2\right)^{-1}}{\sum_j \left(1 + \left\|f_\theta(x_i) - r_j\right\|^2\right)^{-1}} \tag{4}$$

$$p_{ij} = \frac{{q_{ij}}^2 / \sum_i q_{ij}}{\sum_j \left({q_{ij}}^2 / \sum_i q_{ij}\right)} \tag{5}$$

Where x_i is the i-th input and r_j is the j-th cluster representative. q_{ij} is defined as the similarity between the i-th embedded input and j-th cluster representative.

In IDEC, the labels can be obtained from the following formula:

$$s_i = \arg_j max\ q_{i,j} \tag{6}$$

4 Proposed approach: EDCWRN

4.1 Overview

Extracting meaningful cluster-oriented representations is fundamental in deep clustering. By designing the suitable architecture, better representations are extracted, and optimal clusters are formed. To this end, we propose an efficient deep model to learn representations and cluster representatives jointly. Figure 5 shows the architecture of the proposed EDCWRN method. As shown in this figure, the input of this model is a sample and its k nearest neighbors. At the end of training each batch, k nearest neighbors of each sample are calculated. The goal of using neighbors is to learn similar representations for samples within the same cluster. More details on how to calculate and select neighbors are discussed in Subsection 4.3. In the proposed architecture, the encoder module is responsible for extracting useful representations for clustering, and the decoder module uses these extracted representations to calculate cluster labels and reconstruct the input. Observe that in the encoder module, the sample and its neighbors are passed into two separate networks. Representations learned from a sample, and its neighbors are

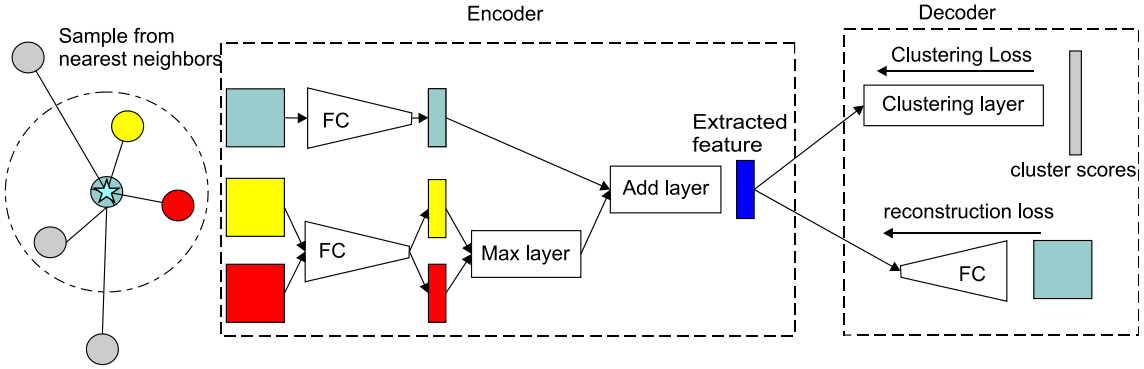

Fig. 5 EDCWRN architecture

summed and fed as input to the decoder module. The main aim of the encoder module is to learn similar representations for samples belonging to the same cluster. This aim is achieved by involving the neighbors in the network training process. As a result, proper embedded feature space is learned for clustering. The decoder module uses the new embedded feature space to reconstruct the input and estimate the clusters. The output of the encoder is fed separately to the fully connected layers (to reconstruct the input) and the clustering layer (to predict clusters). The fully connected layers are the same as the fully connected layers of the encoder module, except that the order of the layers is reversed.

In the proposed architecture, the i-th sample (x_i) passes through fully connected layers, and new representations for the input sample (v_{x_i}) are generated (Eq. (7)).

$$v_{x_i} = \sigma(W_x\, x_i + b_x) \tag{7}$$

where W_x and b_x are the layer parameters (weights and biases) learned during network training, and σ represents the RELU activation function.

Neighbors of the i-th sample (z_i) also pass through fully connected layers and a max layer, and finally, new representations (v_{z_i}) are generated for data neighbors (Eq. (8)).

$$v_{z_i} = \max_{i=1,\ldots,k} \left(\sigma(W_z\,(z_i) + b_z)\right) \tag{8}$$

where, W_z and b_z are the layer parameters (weights and biases) learned during network training, and σ represents the RELU activation function. The weights W_z and b_z are shared among the K neighbors (z_1, z_2, z_3, …, z_k). In other words, the weight sharing technique is used.

Representations generated from a sample (v_{x_i}) and its neighbors (v_{z_i}) are summed and given as input to the decoder module (Eq. (9)).

$$f_\theta(x_i) = v_{x_i} + v_{z_i} \tag{9}$$

where, $f_\theta(x_i)$ is the embedding of the i-th sample.

$f_\theta(x_i)$ is passed separately from the fully connected layers and the clustering layer. The fully connected layers are the same as the fully connected layers of the encoder module, except that the order of the layers is reversed (Eq. (10)).

$$g_\eta \circ f_\theta(x_i) = \sigma(W_x\, f_\theta(x_i) + b_x) \tag{10}$$

In Eq. (10), $g_\eta \circ f_\theta(x_i)$ is reconstructed from the i-th sample obtained from the output layer.

$f_\theta(x_i)$ is also passed to the clustering layer. The weights of this layer are the centers of the clusters, and the output of this layer determines the probability that the i-th sample belongs to the j-th cluster (predicted cluster labels). Eq. (11) shows how to calculate the cluster labels.

$$q_{ij} = \left(1 + a\left(\sum_{m=1}^{M} w_{jm}\left(f_\theta(x_{im}) - r_{jm}\right)^2\right)^b\right)^{-1} \tag{11}$$

where a and b are user-defined positive values, M indicates the number of generated representations (extracted features), w_{km} indicates the weight of the m-th feature in the k-th cluster (for further details about w_{jm} refer to Subsection 4.2), $f_\theta(x_{im})$ indicates the m-th feature in the i-th sample, r_{jm} indicates the m-th feature in the j-th cluster. Unlike the Student's t-distribution (Eq. (2)), the output of this equation is in the range [0,1] and does not need to be normalized.

In designing Eq. (11), inspired by [47], we use the family of curves $(1 + a\, y^{2b})^{-1}$ for modeling cluster predictions. Our proposed equation is not exactly the t-student distribution used in the DEC-based methods, but it is very similar. In the proposed equation (Eq. (11)), unlike the t-student distribution (Eq. (4)), no normalization is applied. Although eliminating the normalization step is an arguably small step for the proposed method, it has a dramatic effect on the performance. This is because summation or integration is a computationally expensive step.

The EDCWRN loss function, like IDEC [22], is a combination of reconstruction loss (L_r) and clustering loss (L_c). Eq. (12) shows the loss function of the proposed method.

$$\mathcal{L} = \mathcal{L}_r + \lambda\, \mathcal{L}_c \tag{12}$$

where λ is a coefficient that controls the degree of distorting embedded space.

The reconstruction loss function ($\mathcal{L}_r$) in the proposed model is the same as the loss function used in IDEC, which is calculated by Eq. (2). In the proposed algorithm, we use Cross-entropy for the clustering loss function, which is calculated between proposed soft computed labels (Eq. (11)) and the target distribution (Eq. (5)). Eq. (13) shows the clustering loss function.

$$\mathcal{L}_c = \sum_i \sum_j p_{ij} \log\left(q_{ij}\right) \tag{13}$$

The proposed EDCWRN algorithm's pseudo-code is given in Fig. 6. As shown in this algorithm, the target distribution P serves as a "ground-truth" soft label but depends on the predicted soft label. Therefore, to avoid instability, P should not be updated at each iteration (one update for autoencoder's weights using a mini-batch of samples is called an iteration) using only a batch of data. In practice, we update target distribution using all embedded points every T iteration. See Eqs. (5) and (11) for the update rules. When updating target distribution, the label assigned to x_i is obtained by Eq. (14).

$$s_i = \arg_j max\ q_{i,j} \tag{14}$$

Where $q_{i,j}$ is computed by Eq. (11). We will stop training if label assignment change (in percentage) between two consecutive updates for target distribution is less than a threshold ε. The optimization proof of EDCWRN is given in the Appendix 1.

In each iteration, a batch of data is selected according to the batch size (Step 9). This batch of data is used to update candidate neighbors (step 10) and network parameters (step 11). In each iteration, the candidate neighbors of each sample in the embedded space (encoder output) are calculated and updated. The nearest neighbor approach is used to calculate candidate neighbors on the batch of data.

4.2 Representation weighting

In existing deep clustering methods, some generated representations are either less important than others or irrelevant to the target problem. However, most deep clustering methods do not consider the importance of the generated representations, and consequently, their performances fall significantly in some applications.

To cope with this problem, we use an automatic local representation weighting strategy to weight the representations of each cluster properly. Representation weighting is performed locally in a way that the extracted representations depending on their importance in the clusters have different weights, hence increasing the quality of clustering. In the proposed local representation weighting strategy, the principle for weighting extracted representations in each cluster that should be taken into account is as follows: a larger weight is assigned to a feature with a smaller variance, and a smaller weight to a feature having a larger variance. According to this principle, the weights of each feature in each cluster are updated simultaneously during network training via Eq. (15).

Fig. 6 The pseudo-code of the EDCWRN algorithm

Input: input data $\boldsymbol{\chi}$, Number of clusters K, Target distribution target interval T, stopping threshold ε, Maximum iterations $MaxIter$

Output: Cluster representatives R, Labels S

1: **for** $iter \in 0, 1, \ldots, Maxiter$
2: **if** $iter \% T == 0$
3: Compute the embeddings for all samples
4: Compute target distribution (P) by Eq. (5)
5: Save last label assignments: $s_{old} = s$
6: Compute new label assignments by Eq. (14)
7: **if** $(sum\ (s_{old} \neq s) / n < \varepsilon)$
8: Stop training
9: Choose a batch of samples $s \in X$
10: Update neighbors on s
11: Update network parameters on s

$$w_{jm} = \frac{1}{\sum_{s=1}^{M} \frac{\sum_{i=1}^{N} {q_{ij}}^{\alpha} \left(f_{\theta}(x_{im}) - r_{jm}\right)^2}{\sum_{i=1}^{N} {q_{ij}}^{\alpha} \left(f_{\theta}(x_{is}) - r_{js}\right)^2}} \tag{15}$$

where N represents the number of samples in a batch, M refers to the number of features (generated representations), w_{km} indicates the weight of the m-th feature in the k-th cluster, $f_{\theta}(x_{im})$ indicates the m-th feature in the i-th sample, and r_{jm} indicates the m-th feature in the j-th cluster. α is the fuzzification coefficient ($\alpha > 1$). For w_{jm} the following constraint is considered (Eq. (16)):

$$w_{j,m} \in [0, 1]. \sum_{m=1}^{M} w_{j,m} = 1. 1 \leq j \leq K. 1 \leq m \leq M; \tag{16}$$

where K is the number of clusters.

4.3 Candidate neighbors

In EDCWRN, we select candidate neighbors using a nearest neighbor approach. We use the embedding feature space (the encoder output) to compute the distance between each pair of samples. In each batch, the candidate neighbors of each sample are calculated. Input samples and their neighbors enter the network. During network training and reaching the final epochs, the better candidate neighbors are selected. The number of neighbors is specified by the user.

5 Experiments

In this section, the results of the experiments are provided, which were carried out to evaluate the proposed method. At first, the datasets which were used in our experiments are presented. Then, in an attempt to demonstrate the effectiveness of our solutions, the results of our extensive experiments are presented. Lastly, the performance of this proposed method is assessed on testing datasets and compared with the following state-of-the-art methods:

- deep adversarial subspace clustering (DASC) [48],
- deep embedded clustering (DEC) [27],
- variational deep embedding (VaDE) [42],
- joint unsupervised learning (JULE) [50],
- deep clustering with convolutional autoencoders (DCEC) [53]
- deep clustering for unsupervised learning of visual features (DeepCluster) [45]
- deep embedded regularized clustering (DEPICT) [46],
- improved deep embedded clustering with locality preservation (IDEC) [22],
- deep spectral clustering using a dual autoencoder network (DSCDAN) [47].
- discriminatively boosted clustering (DBC) [27]
- deep k –Means (DKM) [48]
- semi-supervised deep embedded clustering (SDEC) [49]
- adversarial deep embedded clustering (ADEC) [30]
- deep clustering with a dynamic autoencoder (DynAE) [23]
- large-scale multi-view subspace clustering (LMVSC) [50]
- scalable deep k-subspace clustering (SDKSC) [41]
- deep embedding clustering based on contractive autoencoder (DECCA) [24],
- deep clustering network (DCN) [43],
- pseudo-supervised deep subspace clustering (PSSC) [53]

similar to the settings in IDEC, the encoder module is a set of dense layers (fully connected layers) with dimensions d _ 500 _ 500 _ 2000 _ 10, where d is the dimension of input data, and the decoder module is the same as the fully connected layers of the encoder module, except that the order of the layers is reversed (10 _ 2000 _ 500 _ 500 _ d). The entire internal layers are activated using the ReLU nonlinearity function. The autoencoder network pre-training is adjusted as [27]. For more details, refer to the paper. The clustering loss coefficient λ is set to 0.1, and the number of neighbors is set to 2. Parameter batch size, the update intervals T, and the convergence threshold ε are set to 256, 140, and 0.1%, respectively. SGD (Stochastic Gradient Descent) optimizer with learning 0.1 and momentum 0.9 is used in the EDCWRN framework. The implementation- source code- of EDCWRN is made publicly accessible at https://github.com/Amin-Golzari-Oskouei/EDCWRN.

In the pertaining and training phases, we use data augmentation approaches. We use five transformations that are randomly applied to samples. The details of these transformations are summarized in Table 1.

5.1 Dataset

To closely investigate the effects of the strategies presented in our method and compare the proposed method with other methods, we use six benchmark datasets. The statistic of the datasets is given in Table 2.

- ***Mnist***: This dataset consists of 28 by 28 Gy-scale images of 70,000 handwritten digits.
- ***Fashion-Mnist***: This dataset consists of 28 by 28 Gy-scale images of 10 fashion categories.

Table 1 Transformations

	Width shift	Height shift	Rotation	Zoom
Range	0.1	0.1	10	0.1

Table 2 Datasets

Datasets	No. of Samples	No. of Classes	No. of Dimensions	Type
Mnist	70,000	10	784	Image
Fashion-Mnist	70,000	10	784	Image
Mnist-Test	10,000	10	784	Image
USPS	9298	10	256	Image
Reuters-10 k	10,000	4	2000	Text
20NG	18,846	20	2000	Text

- ***Mnist-Test*:** This dataset is a test set of the *Mnist* dataset, containing 28 by 28 Gy-scale images of 10,000 handwritten digits.
- ***USPS*:** This dataset is a handwritten digit from the *USPS* postal service. It contains 9298 samples of 16 by 16 images.
- ***Reuters-10 k*:** Following the IDEC model [22], we randomly sampled a subset of 10,000 examples and computed TFIDF features on the 2000 most frequent words from the REUTERS dataset. We used four root categories: corporate/industrial, government/social, markets, and economics as labels.
- ***20NG*:** This dataset consists of 18,846 text documents. These documents are partitioned into 20 different groups according to their topics. We compute TFIDF features on the 2000 most frequent words from the *20NG* dataset.

5.2 Evaluation criteria

In our experiments, the clustering performances of the different methods are evaluated concerning two standard measures: *Normalized Mutual Information (NMI) and Accuracy (ACC).*

NMI NMI is an information-theoretic measure based on the mutual information of the ground-truth classes and the clusters obtained from a clustering algorithm [54], defined as Eq. (17):

$$NMI\ (R,Q) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J} P(i,j)\ log\frac{P(i,j)}{P(i)\ P(j)}}{\sqrt{H(R)H(Q)}}, \tag{17}$$

where R and Q are two partitions belonging to the input dataset and include I and J clusters, respectively. $P(i)$ is the probability that a randomly selected sample from the input data assigns to a cluster R_i, $P(i, j)$ is the probability referring to a sample that belongs to both clusters R_i and Q_i. $H(R)$ is the entropy relating to all the probabilities $P(i)$ $(1 \leq i \leq I)$ in partition R.

Accuracy Different from *NMI*, *ACC* measures the proportion of data points for which the obtained clusters can be correctly mapped to ground-truth classes [55], calculated as Eq. (18):

$$ACC = \frac{\sum_{k=1}^{K} d_k}{N}, \tag{18}$$

where d_k indicates the number of data points that are correctly assigned in the k-th cluster, and N indicates the number of all samples.

5.3 Experiment 1: The effect of representation weighting, neighboring, and proposed formulation

To study the effect of representation weighting, neighboring, and proposed formulation implemented in our method on the final results, we evaluate the performance of the proposed model using these techniques and without them. Table 3 presents the *Accuracy* and *NMI* rates from top to bottom for each dataset, respectively. In the Representation weighting and Neighboring columns, the symbol "✓" indicates the use of that technique, and the symbol "✗"indicates that the technique is not used. Also, in the Formulation column, the word "proposed" is used if the formulation is based on the proposed equations (Eqs. (11) and (13)), and the word "IDEC" is used if the formulation is based on the proposed equations in IDEC (Eqs. (3) and (4)).

Discussion on representation weighting As shown in Table 3, the performance of the proposed approach with the representation weighting technique is better than without the weighting mode. When the local representation weighting is used, the *ACC* and *NMI* rates of the proposed approach improve by an average of 0.58% and 0.95% on all testing datasets, respectively. The effect of the representation weighting technique is significant in some datasets, such as *Fashion-Mnist*, *Mnist*, *20NG*, and *Reuters-10 k*.

In the *Fashion-Mnist* dataset, the *ACC* and *NMI* rates of the proposed approach with representation weighting modes are improved by an average of 1.58% and 1.68%, respectively, as opposed to without representation weighing modes. Also, in the *Mnist* dataset, the *ACC* and *NMI* rates of the proposed approach with representation weighting modes, as opposed

Table 3 Effect of representation weighting, neighbors, and proposed formulation

Representation weighting	Neighboring	Formulation	Dataset					
			Mnist	*Fashion-Mnist*	*Mnist-Test*	*USPS*	*Reuters-10 k*	*20NG*
✓	✓	proposed	**98.80**	**62.50**	**98.66**	**98.21**	**83.72**	**58.14**
			96.60	**68.09**	**96.35**	**95.18**	**60.64**	**51.83**
✓	✓	IDEC	98.66	62.36	98.58	97.78	82.91	57.26
			96.31	67.80	96.19	94.39	59.84	51.26
✓	✘	proposed	98.69	62.40	98.54	97.63	83.45	57.75
			96.39	67.88	96.07	94.45	60.31	51.35
✓	✘	IDEC	98.72	62.44	98.59	97.34	82.80	57.20
			96.44	68.03	96.23	93.81	59.64	51.20
✘	✓	proposed	98.72	62.46	98.60	98.00	82.85	57.30
			96.47	68.08	96.19	94.65	59.66	51.31
✘	✓	IDEC	94.78	58.56	98.64	97.76	81.97	57.20
			88.36	63.41	96.34	94.48	58.93	51.14
✘	✘	proposed	98.61	62.36	98.51	97.60	82.06	57.24
			96.23	67.84	96.00	94.36	59.11	51.23
✘	✘	IDEC	98.66	62.42	98.59	97.34	81.72	56.14
			96.35	67.96	96.23	93.81	58.64	49.83

The best results are in boldfaced

to without representation weighting modes, are improved by an average of 1.04% and 2.20%, respectively. This indicates that in these datasets, the new representation space is not sufficiently suitable for clustering. In contrast, in some datasets, such as *Mnist-Test*, the representation weighting technique does not have a significant effect on results. When the data in the new embedded space is suitable for clustering, the effect of representation weighting is reduced. For a better understanding, the weights obtained for each dataset are illustrated in Fig. 7. As shown in this figure, in datasets such as *USPS* and *Mnist-test*, the weights obtained in each cluster are approximately balanced. In contrast, for *Mnist* and *Fashion-Mnist* datasets, these weights are unbalanced. In the latter case, the weighting of representations has a remarkable impact on optimal cluster formation. The difference in weights obtained in each cluster indicates that the importance of representations in the clustering process is not equal.

For almost all clusters of the *20NG* dataset, the zero to third features are more important than the other attributes (see Fig. 7f). Of course, the degree of importance of each feature in each cluster is different. Table 3 also shows that weighting the features increases clustering accuracy. Evaluation of the unbalanced *Reuters-10 k* dataset also shows that some features are very important in some clusters, while these features may be less important in some other clusters. For example, in cluster 1, the zero feature has a high significance, while in cluster 3, this feature is less important.

Discussion on neighbors As shown in Table 3, the proposed approach performs better when using the neighboring technique than the one which doesn't use it. By using the data neighbors, better representations are generated. For the *Mnist-test* dataset, the *ACC* and *NMI* rates of the proposed approach (first row in Table 3), compared to without using neighbor modes, are improved by an average of 0.10% and 0.22%, respectively. Also, for the *USPS* dataset, the *ACC* and *NMI* rates of the proposed approach are improved by an average of 0.75% and 1.13%, respectively, as opposed to without using neighbor modes. Similarly, for the *Mnist* and *Fashion-Mnist* datasets, the results in the proposed method are better than in the case without the use of neighbors. The same discussion is also true for the *Reuters-10 k* and *20NG* datasets. The neighboring technique effectively improves network performance. However, the selection of neighbors depends on the learned embedded space. If the suitable embedded space is not learned for clustering, the neighbors will not be selected correctly, and as a result, the network performance may be reduced. The two steps are essential for generating better cluster-oriented embedding space: (1) pre-training phase and (2) designing appropriate architecture based on the data type. If the network pre-training is done well, from the initial epochs of training, appropriate and correct neighbors are selected, and as a result, better representations are generated. By applying both of them, a suitable embedded space will be generated. Therefore, better neighbors are selected. In the proposed architecture, inspired by IDEC, we use dense layers. Certainly,

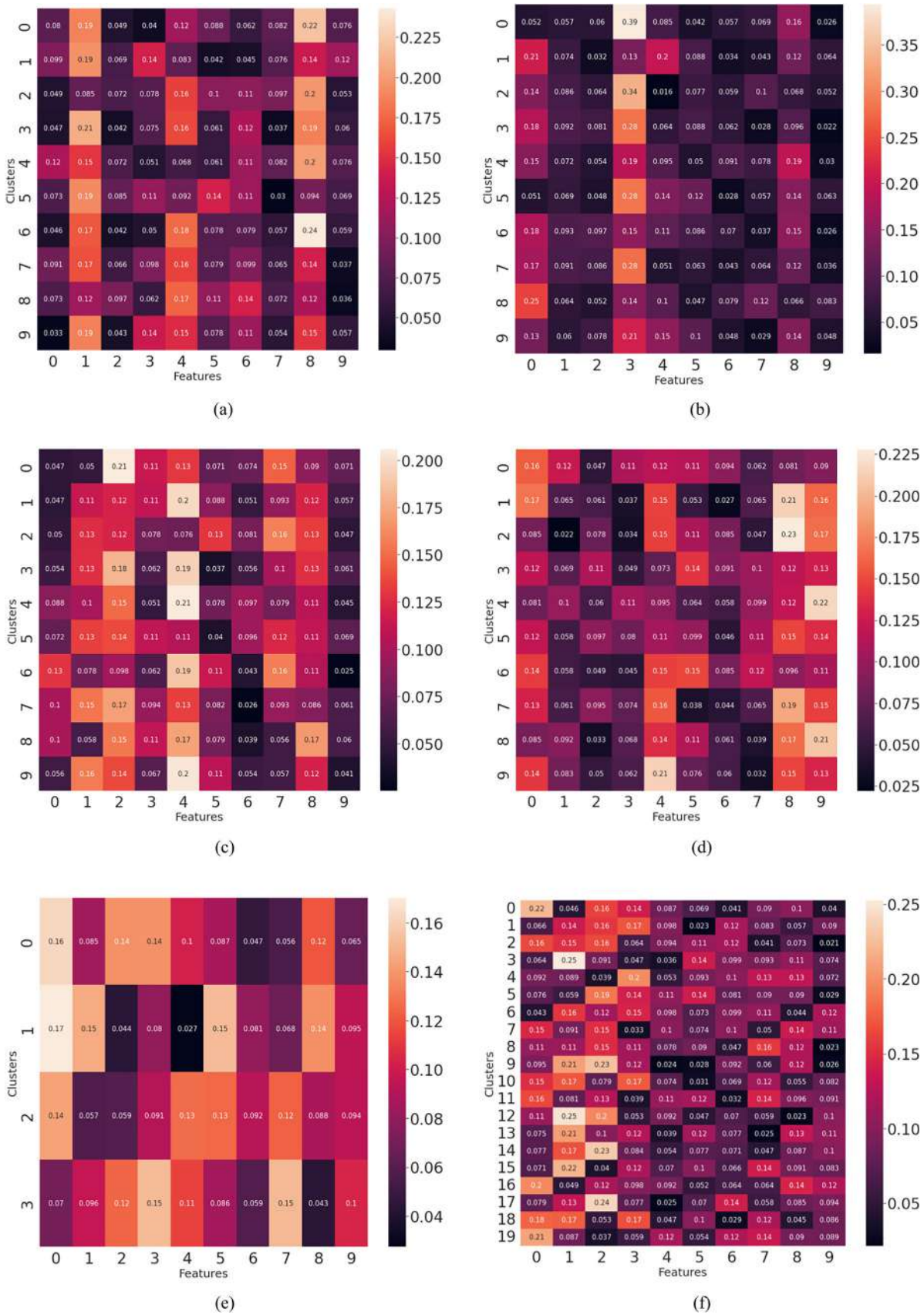

Fig. 7 Visualization of the weight of representations. (**a**) *Mnist*, (**b**) *Fashion-Mnist*, (**c**) *Mnist-Test*, (**d**) *USPS*, (**e**) *Reuters-10 k*, and (**f**) *20NG*

simple dense layers are not suitable for generating the proper representations for the tested image datasets. The purpose is to compare fairly with other methods and investigate the effect of each used technique.

Discussion on formulation As shown in Table 3, the proposed approach performs better with the proposed formulation. Using a new formation, the *ACC* and *NMI* rates of the proposed method on all testing datasets are improved by an average of 0.72% and 1.19%, respectively. For some datasets, such as *USPS* and *Fashion-Mnist*, the effect of the proposed formulation is more significant than other datasets. In the *Mnist* dataset, the *ACC* and *NMI* rates of the proposed approach are improved by an average of 1.02% and 2.18%, respectively, as opposed to IDEC modes. Similarly, in the *Fashion-Mnist* dataset, the *ACC* and *NMI* rates of the proposed approach, compared to IDEC modes, are improved by an average of 1.60% and 1.75%, respectively. Also, for the *USPS* and *Mnist-test* datasets, the proposed method has better results than the IDEC modes.

As shown above, all used techniques somehow positively affect the final result. Representation weighting, proposed formulation, and applied neighboring techniques have the most positive effect on the final results, respectively.

5.4 Experiment 2: Visualization of the learned representations

In an attempt to offer a more interpretable outlook of the representations generated through deep clustering algorithms, we visualize the embedded samples generated by the EDCWRN, DEC [27], IDEC [22], DCEC [53], and DynAE [23] on all tested datasets. We use the UMAP (Uniform Manifold Approximation and Projection) [47] visualization approach to project the embeddings into a 2D space. Some comparative methods were developed only on a specific task, such as image or text. Therefore, comparisons are performed on image and text datasets separately to investigate the performance of the algorithms.

5.4.1 Image datasets visualization

Figures 8, 9, 10 and 11 show the visualization of different methods on *Mnist*, *Fashion-Mnist*, *Mnist-Test*, and *USPS* datasets, respectively. As shown in these figures, the representations for samples from different clusters are better separated and disentangled in EDCWRN than in the other methods. This further validates our experimental results, indicating the superior ability of EDCWRN to learn representations that improve clustering. To better evaluate each of the algorithms and compare their results with the proposed method, EDCWRN is compared with each of the algorithms one by one:

EDCWRN vs. DEC For the *Mnist* and *Mnist-Test* datasets, as shown in Figs. 8 and 9, the proposed method can well preserve the distance between and within the cluster. In contrast, the DEC method can not distinguish the two clusters well. Thus, almost all samples of these two clusters are incorrectly predicted. For the *USPS* and *Fashion-Mnist* dataset datasets, in the proposed method, the intra-cluster variance is less than the DEC method, which indicates that the clusters are more compact.

EDCWRN vs. IDEC Similar to the DEC method, the IDEC method can not distinguish some clusters well. This is obvious in the *Mnist*, *Mnist*-Test, and *USPS* datasets (see Figs. 8, 10, and 11). While for these datasets, the EDCWRN method can distinguish the clusters well. Also, using the IDEC method, almost none of the clusters of the *Fashion-Mnist* dataset don't separate well (see Figs. 9). However, in the EDCWRN, the variance between the clusters and within the cluster is relatively preserved.

EDCWRN vs. DynAE DynAE creates better clusters than the other two previous methods. This method works well for *Mnist*, *Mnist*-Test, and *USPS* datasets (see Figs. 8, 10, and 11). For these datasets, the global (inter-cluster) distance is almost preserved. However, in terms of intra-cluster variance, the proposed method performs better. For the fashion-*Mnist* dataset, the DynAE method does not work properly. So that almost none of the clusters will be separated.

EDCWRN vs. DCEC: using the DCEC method, a few clusters of the *Fashion-Mnist* dataset separated well. Also, in some datasets, samples of a cluster are concentrated at two or more points. This is obvious in the *Mnist*, *Mnist*-Test, and *USPS* datasets. This is quite clear in the yellow cluster of the *Mnist* dataset. In contrast, EDCWRN performs better than the DCEC algorithm because the samples do not overlap, and the variance between and within the cluster is further preserved than in the DCEC method.

5.4.2 Text datasets visualization

Figures 12 and 13 show the visualization of different methods on *Reuters-10 k*, and *20NG* datasets, respectively. As shown in these figures, the representations for samples from different clusters are better separated and disentangled in EDCWRN than in the other methods. This further validates our experimental results, indicating the superior ability of EDCWRN to learn representations that improve clustering. To better evaluate each of the algorithms and compare their results with the proposed method, EDCWRN is compared with each of the algorithms one by one:

EDCWRN vs. DEC For the *Reuters-10 k* dataset, as shown in Fig. 12, the DEC method can separate the clusters more

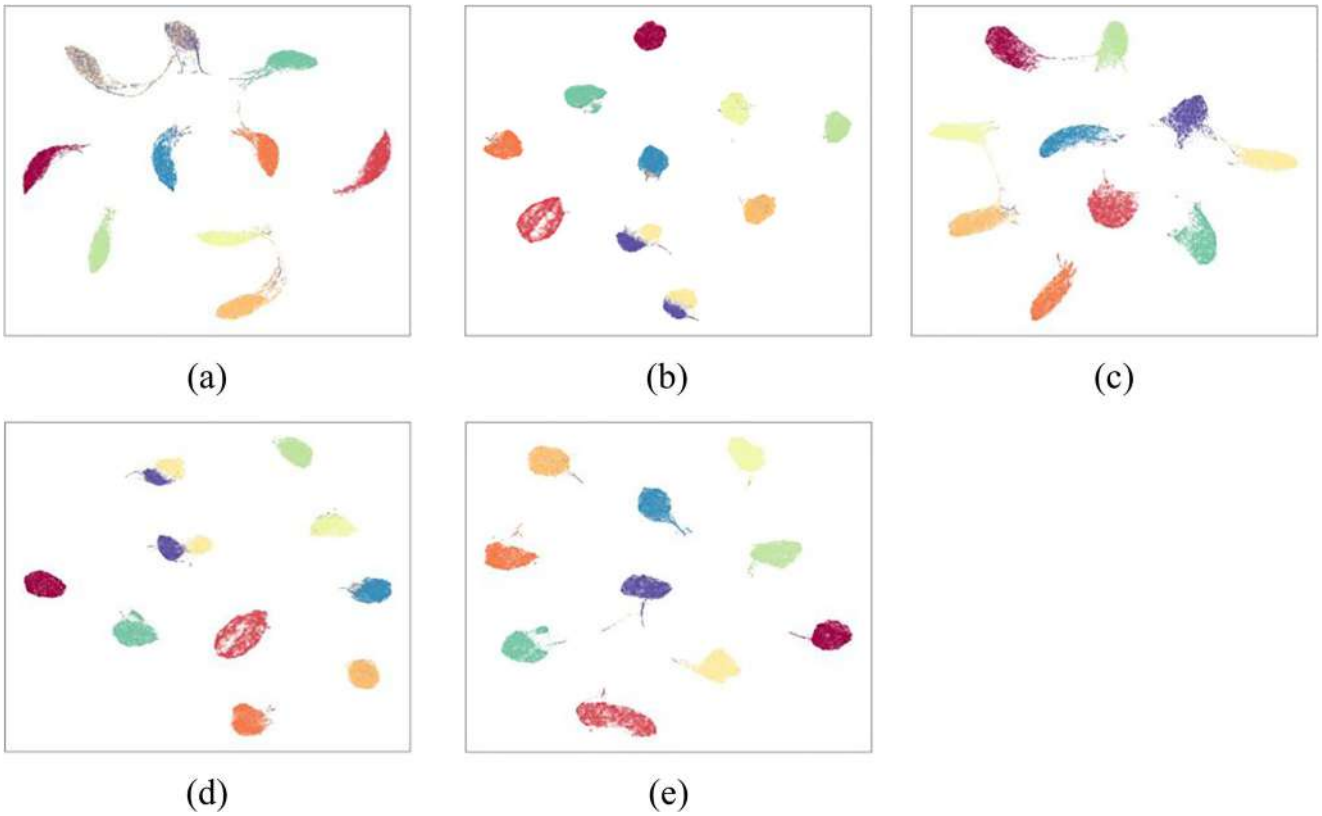

Fig. 8 Visualization of the learned representations on *Mnist* dataset. **a** DEC, **b** IDEC, **c** DynAE, **d** DCEC, and **e** EDCWRN

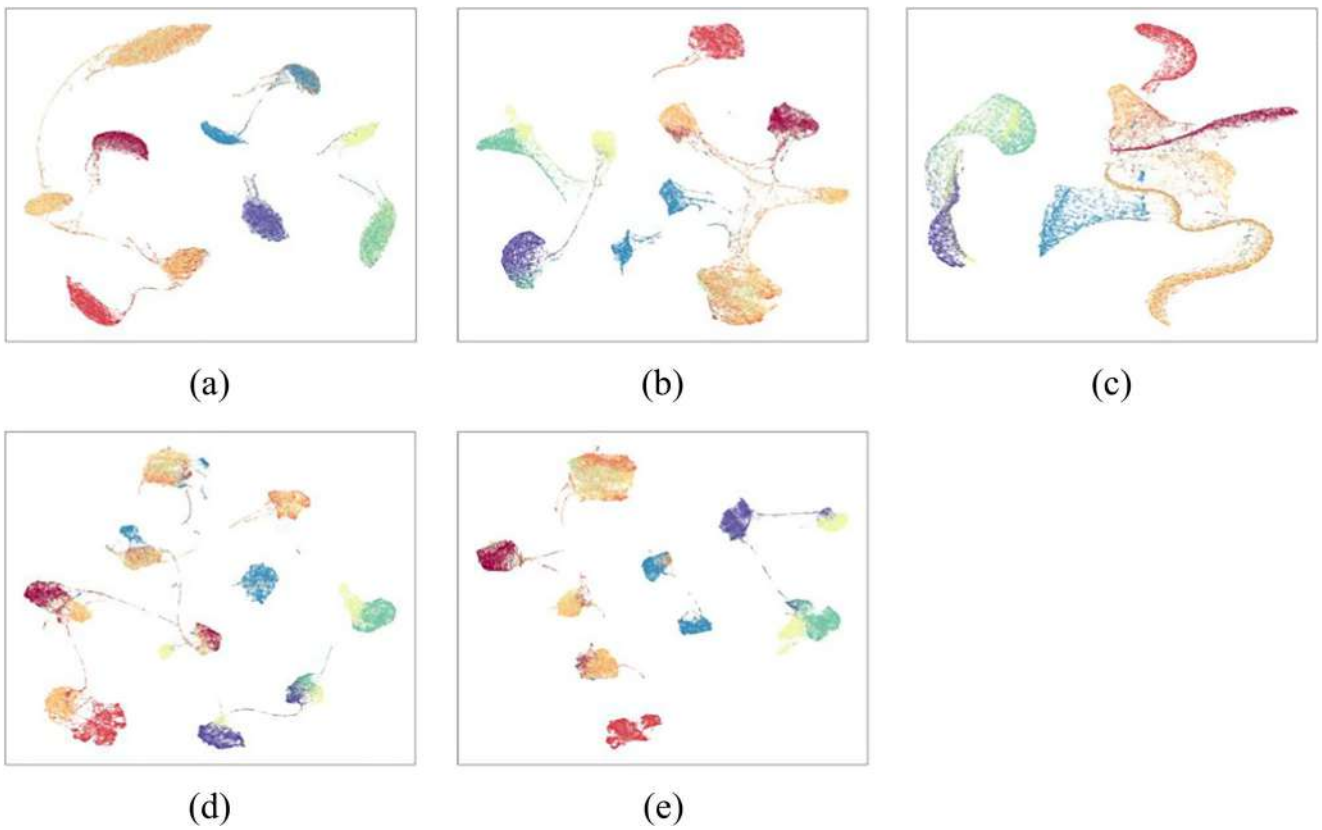

Fig. 9 Visualization of the learned representations on *Fashion-Mnist* dataset. **a** DEC, **b** IDEC, **c** DynAE, **d** DCEC, and **e** EDCWRN

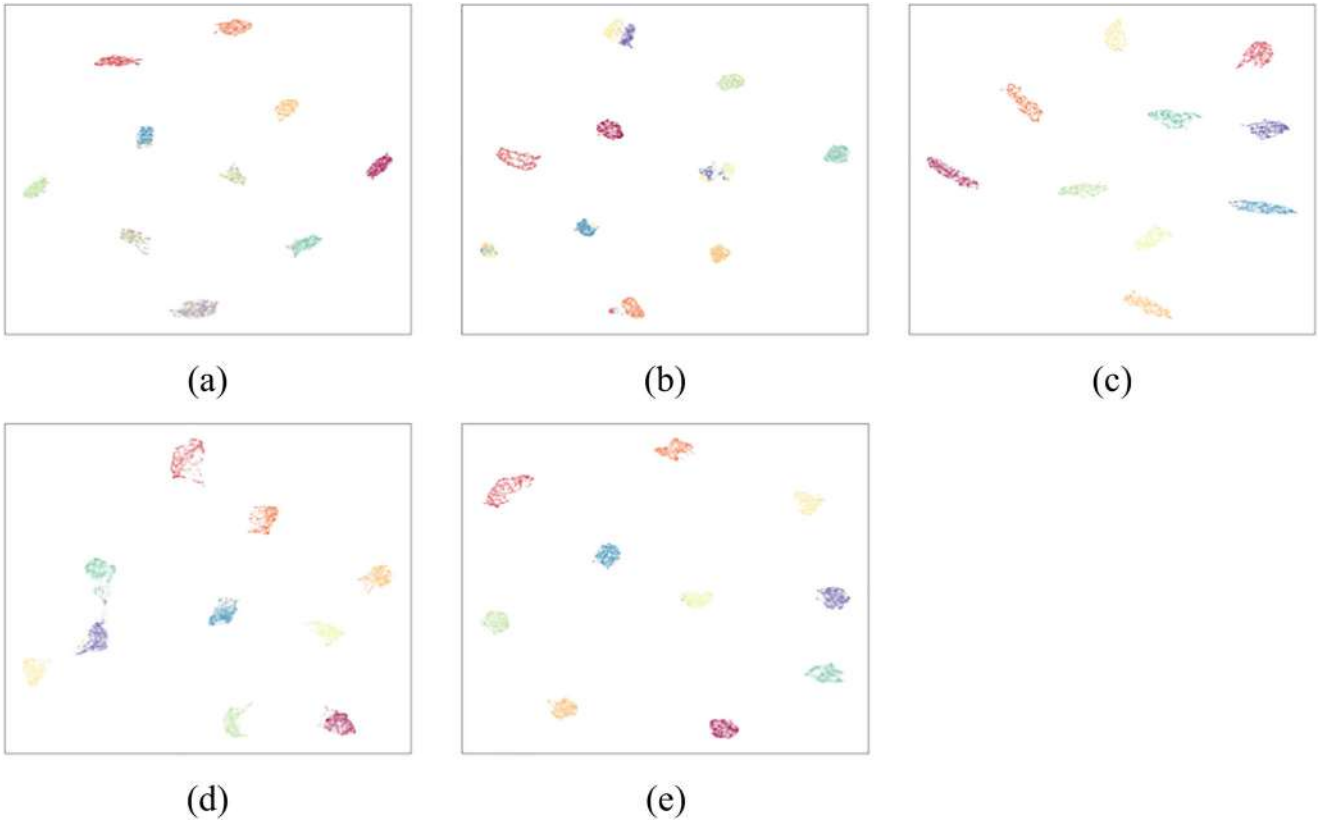

Fig. 10 Visualization of the learned representations on *Mnist-Test* dataset. **a** DEC, **b** IDEC, **c** DynAE, **d** DCEC, and **e** EDCWRN

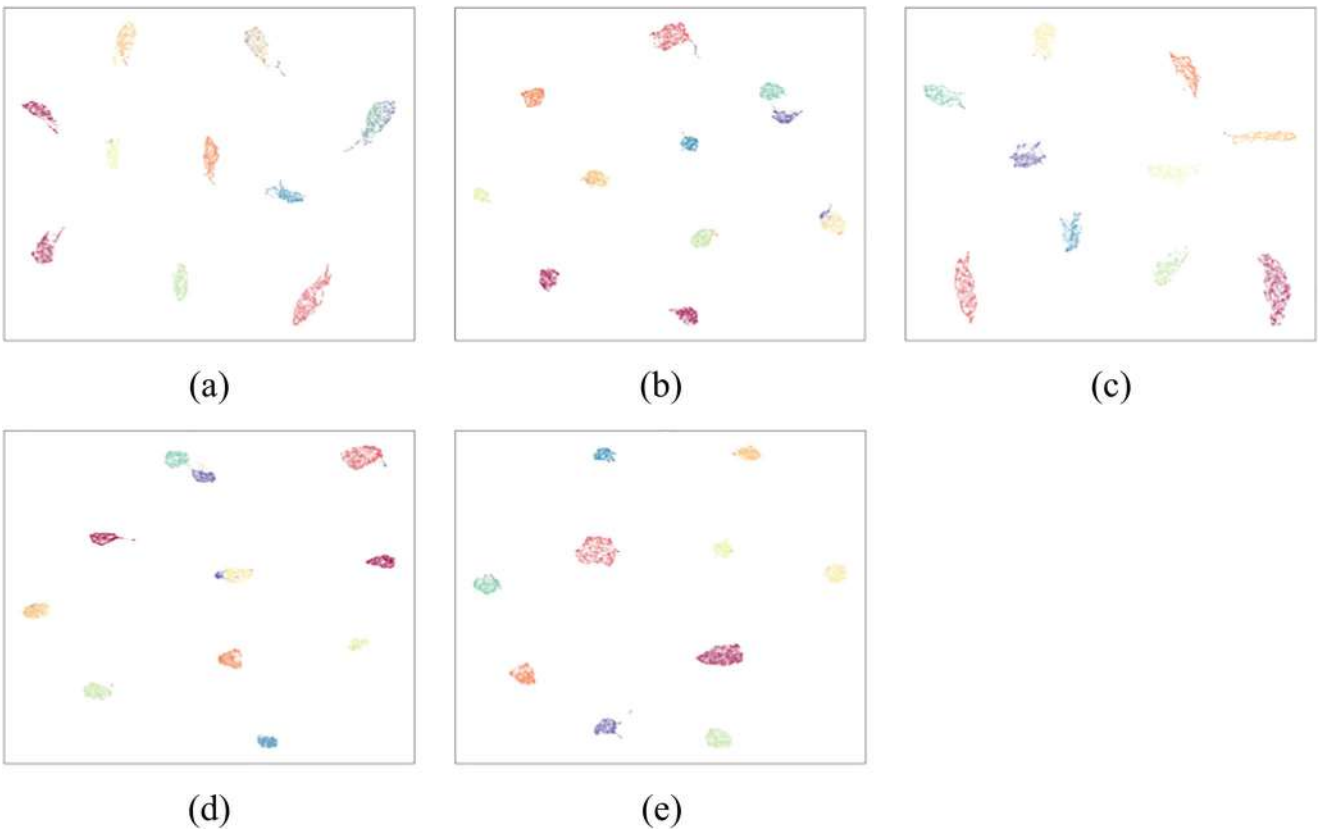

Fig. 11 Visualization of the learned representations on *USPS* dataset. **a** DEC, **b** IDEC, **c** DynAE, **d** DCEC, and **e** EDCWRN

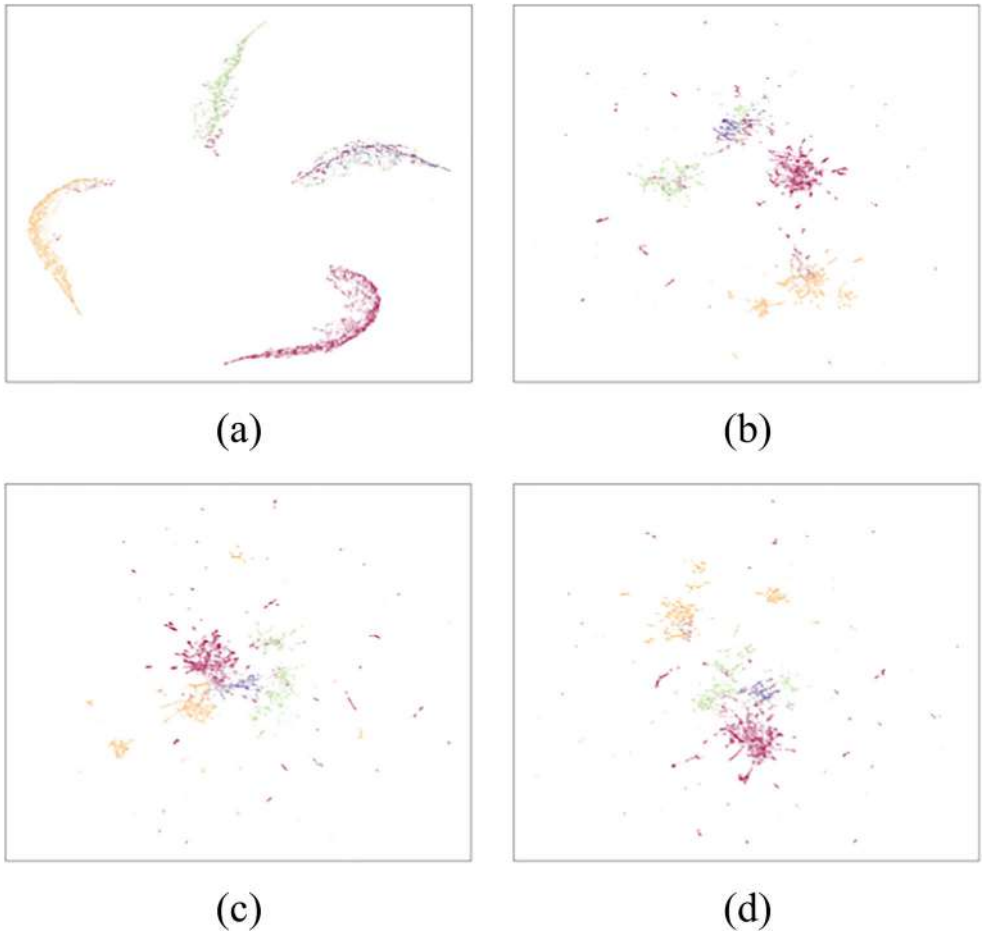

Fig. 12 Visualization of the learned representations on *Reuters-10 k* dataset. **a** DEC, **b** IDEC, **c** DynAE, and **d** EDCWRN

distinctly. But a closer look reveals that in some clusters, almost all samples are incorrectly predicted. This is also true for the *20NG* data set. Although the proposed method does not preserve the inter-cluster distance for the Reuters-10 dataset, it performs better than the DEC algorithm because the samples do not overlap.

EDCWRN vs. IDEC Similar to the DEC method, the IDEC method can not distinguish some clusters well. This is obvious in the *20NG* dataset (see Figs. 13), while the EDCWRN method can determine the clusters well. Also, using the IDEC method, some of the clusters of the *Reuters-10 k* dataset don't separate well (see Figs. 12). However, on this dataset, EDCWRN performs better than the IDEC algorithm because the samples do not overlap.

EDCWRN vs. DynAE For the *Reuters-10 k* dataset, as shown in Fig. 12, it appears that the DynAE method can not well preserve the distance between the clusters. Also, on the *20NG* dataset, almost all the samples are overlapped, and clusters don't well separate. However, on these datasets, EDCWRN performs better than the DynAE algorithm because the samples do not overlap, and the variance between and within the cluster is further preserved than DynAE.

5.5 Experiment 3: Analysis of the behavior of the loss function

To investigate the effect of the loss function adopted in the proposed method and compare it with the KLD loss function, we evaluate the proposed method once with the cross-entropy loss function (Eq. (13)) with the proposed distribution (Eq. (11)) and once with the KLD loss function (Eq. (3)) with T student distribution (Eq. (4)). In both cases, the representation weighting and neighboring techniques are used. In this experiment, we calculate the inter-cluster variance in the embedded space for both cases. Table 4 shows the results.

As shown in Table 4, for all datasets, the proposed method using cross-entropy with the proposed distribution has a high inter-cluster variance. In contrast, the proposed method using KLD with T student distribution has a low inter-cluster variance. The results show that for the *Mnist* dataset, the inter-cluster variance in the proposed method is much higher than in the opposite case. After the *Mnist* dataset, *Mnist-Test*, *USPS,* and *Fashion-Mnist* have the most inter-cluster variance, respectively. Also, for the text dataset, the variance in the proposed method is slightly higher. These results show that the proposed method using the cross-entropy loss function with

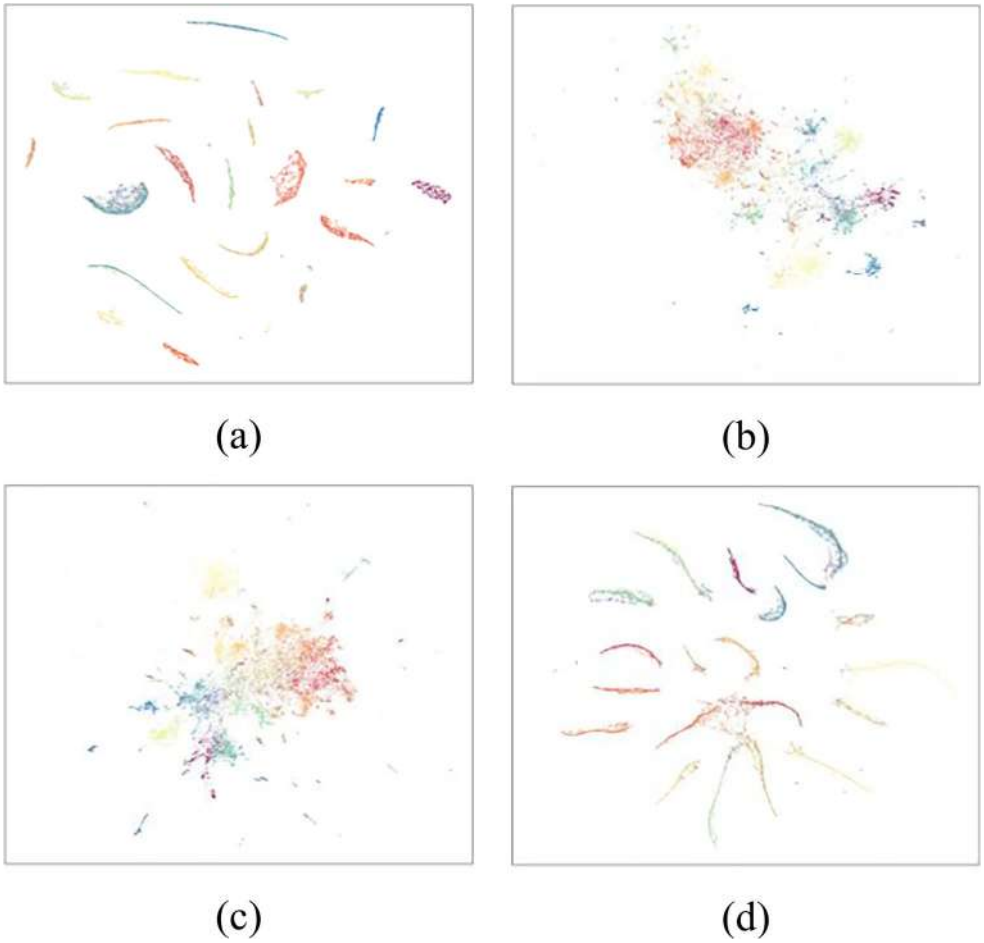

Fig. 13 Visualization of the learned representations on *20NG* dataset. **a** DEC, **b** IDEC, **c** DynAE, and **d** EDCWRN

the proposed distribution can preserve the inter-cluster distance (global distance) much better than the opposite.

5.6 Experiment 4: Analysis of the robustness to noise

To investigate the robustness of the proposed method and compare it with other methods, in this experiment, we add three common kinds of noise to the image datasets as follows: Gaussian noise with the variance of 0.01, 0.03, and 0.05; Salt&Pepper noise with density 0.05, 0.1, and 0.2; Speckle noise with variance 0.05, 0.1, and 0.15. The results of the EDCWRN, the DEC [27], IDEC [22], and DynAE [23] methods on the noisy data are shown in Table 5. The table provides the *ACC* and *NMI* rates from top to bottom for each method, respectively.

As shown in Table 5, the proposed method performs better than other methods for all noise levels and datasets. The difference between the best and worst results obtained on all datasets in the proposed method is on average 1.15%, and 1.17% for *ACC* and NMI metrics, respectively. For the DEC algorithm, this value is 4.07% (*ACC*) and 3.05% (NMI); for the IDEC algorithm, this value is 2.25% (*ACC*) and 2.15% (NMI); and for the DynAE algorithm, this value is 1.35%

Table 4 Comparison of the inter-cluster variance of EDCWRN with different loss functions

Method	Dataset					
	Mnist	*Fashion-Mnist*	*Mnist-Test*	*USPS*	*Reuters-10 k*	*20NG*
KLD loss function with T student distribution	1.842	0.73	0.57	0.699	0.428	1.469
cross-entropy loss function with the proposed distribution	**10.69**	**1.66**	**2.539**	**2.41**	**0.43**	**1.569**

The best results are in boldfaced

Table 5 Clustering performance with different kinds of noise

Dataset	Method	Noise								
		Gaussian			Salt&Pepper			Speckle		
		(0.01)	**(0.03)**	**(0.05)**	**(0.05)**	**(0.1)**	**(0.2)**	**(0.05)**	**(0.1)**	**(0.15)**
Mnist	**DEC**	85.1 82.2	85.0 81.9	84.9 81.8	84.2 81.7	83.5 81.2	81.2 80.0	84.5 81.0	83.9 81.2	82.2 80.5
	IDEC	88.1 86.7	88.0 86.5	87.9 86.0	87.9 86.3	87.0 86.1	85.7 84.9	87.7 86.2	87.5 86.3	86.9 86.0
	DynAE	98.7 96.4	98.6 96.3	98.5 96.3	98.5 **96.6**	98.0 96.4	97.5 95.2	98.3 96.5	98.1 96.5	97.9 96.3
	EDCWRN	**98.8** **96.6**	**98.7** **96.5**	**98.6** **96.5**	**98.6** 96.5	**98.1** **96.4**	**97.8** **95.6**	**98.6** **96.5**	**98.2** **96.5**	**98.0** **96.4**
Fashion-Mnist	**DEC**	51.7 54.4	51.6 54.1	51.5 54.0	50.7 53.9	50 53.4	47.7 52.2	51.0 53.2	50.4 53.4	48.7 52.7
	IDEC	52.8 55.6	52.5 55.4	52.4 54.9	52.4 55.2	51.5 55.0	50.2 52.7	52.2 55.1	52.0 55.5	51.4 54.9
	DynAE	59.0 64.0	58.9 63.9	58.8 63.7	58.8 63.3	58.3 63.0	57.3 62.4	58.6 63.6	58.4 63.2	58.2 63.0
	EDCWRN	**62.4** **67.9**	**62.3** **67.8**	**62.2** **67.7**	**62.3** **67.8**	**61.7** **67.2**	**61.1** **66.4**	**62.0** **67.5**	**61.7** **67.2**	**61.5** **66.9**
Mnist-test	**DEC**	84.9 81.8	84.7 81.6	84.2 81.0	83.4 80.3	82.6 80.1	80.4 79.7	83.8 80.9	83.1 80.2	81.3 80.0
	IDEC	84.5 80.0	84.6 79.8	84.5 77.8	84.5 79.6	83.6 79.4	82.3 78.2	84.3 79.5	84.1 79.6	83.5 79.3
	DynAE	98.6 96.3	**98.5** 96.3	98.5 96.2	98.4 96.6	98.0 96.3	97.4 95.2	98.3 96.4	98.0 96.3	97.8 96.2
	EDCWRN	**98.6** **96.5**	98.4 **96.5**	**98.5** **96.4**	**98.5** **96.6**	**98.1** **96.4**	**97.6** **95.4**	**98.5** **96.4**	**98.1** **96.5**	**97.9** **96.4**
USPS	**DEC**	76.0 76.4	75.8 76.0	75.6 75.7	75.1 75.9	74.4 75.4	72.1 74.2	75.2 75.2	74.8 75.4	73.1 74.7
	IDEC	76.1 78.5	76.0 78.0	75.9 77.5	75.9 77.8	75 77.6	73.7 76.4	75.7 77.7	75.5 77.8	74.9 77.5
	DynAE	98.1 94.8	98.0 94.7	97.9 94.7	97.9 95.0	97.4 94.8	96.9 93.6	97.7 94.9	97.5 **94.9**	**97.3** **94.7**
	EDCWRN	**98.3** **95.2**	**98.2** **95.1**	**98.2** **95.1**	**98.2** **95.1**	**97.6** **94.9**	**97.0** **94.1**	**98.2** **95.0**	**97.5** 94.7	97.2 94.3

The best results are in boldfaced

(*ACC*) and 1.27% (NMI). As the statistical results show after EDCWRN, the DynAE, IDEC, and DEC methods have the best results, respectively. Therefore, the proposed algorithm has a better overall performance than the other methods on all datasets and is more robust to noise.

5.7 Experiment 5: EDCWRN vs. state-of-the-art methods

In this section, EDCWRN's performance is compared with other state-of-the-art methods. Some comparative methods were evaluated only on a specific dataset, such as image or text. Therefore, comparisons are performed on image and text datasets separately to investigate the performance of the algorithms.

5.7.1 Evaluation of image datasets

In this section, we compare the performance of the proposed algorithm with other clustering methods on image datasets. The performance of different methods is illustrated in Table 6. The table provides the *ACC* and *NMI* rates from top to bottom for each method, respectively. The results related to other methods have been mentioned directly from the relevant publications.

As Table 6 shows, the performance of EDCWRN is the best concerning all datasets except *Fashion-Mnist*. The results indicate the higher performance of EDCWRN compared to other successful methods in this field. After EDCWRN, DynAE, ADEC, and DSCDAN methods have relatively good performance, respectively. Also, we can find that almost methods perform well on all

Table 6 The performances of the proposed algorithm and other clustering methods

Methods	Datasets			
	Mnist	*Fashion-Mnist*	*Mnist-Test*	*USPS*
DEC [25]	86.3	51.8	85.6	76.2
	83.4	54.6	83.0	76.7
DCN [52]	83.0	50.1	80.2	73.0
	81.0	55.8	78.6	71.9
PSSC [53]	89.0	–	–	93.5
	79.0	–	–	85.6
DASC [41]	80.1	–	80.4	–
	78.4	–	78.0	–
VaDE [42]	94.5	57.8	28.7	56.6
	87.6	63.0	28.7	51.2
DCEC [44]	88.9	49.0	85.29	79
	88.4	51.9	83.61	82.5
JULE [43]	96.4	56.3	96.1	95.0
	91.3	60.8	91.3	91.3
SDEC [49]	86.11	52.8	–	76.3
	82.89	59.31	–	77.6
DEPICT [46]	96.5	39.2	96.5	96.4
	91.7	39.2	91.5	92.7
IDEC [22]	88.1	52.9	84.6	76.1
	86.7	55.7	80.2	78.5
DSCDAN [47]	97.8	**66.2**	98.0	86.9
	94.1	64.5	94.6	85.7
DBC [27]	96.4	–	–	74.3
	91.7	–	–	72.4
DKM [48]	86.2	–	–	77.0
	80.5	–	–	78.7
ADEC [30]	98.6	58.6	98.5	98.1
	96.1	66.2	95.7	94.8
DynAE [23]	98.7	59.1	98.7	98.1
	96.4	64.2	96.3	94.8
SDKSC [51]	87.1	60.0	83.3	–
	78.1	62.3	77.4	–
DeepCluster [45]	79.7	54.2	85.4	56.2
	66.1	51.0	71.3	54.0
DECCA [24]	96.3	60.9	–	77.31
	90.7	66.9	–	80.53
LMVSC [50]	55.7	51.2	57.2	67.5
	50.3	51.9	51.6	62.1
EDCWRN	**98.8**	62.5	**98.7**	**98.3**
	96.6	**68.0**	**96.3**	**95.2**

The best results are in boldfaced

image datasets. What is more, EDCWRN outperforms traditional deep clustering algorithms, such as DEC, IDEC, and DCN, with a large margin. That indicates the fascinating potential of EDCWRN in the unsupervised clustering field.

The average results for EDCWRN and other compared algorithms are shown in Table 7. As this table shows, EDCWRN has the best results. Regarding *ACC* criteria, after EDCWRN, the methods of DynAE, ADEC, JULE, and DEPICT have better outcomes, respectively. Also, in terms of NMI criteria, after EDCWRN, the methods ADEC, DynAE, JULE, and DEPICT have better outcomes, respectively.

5.7.2 Evaluation of text datasets

In this section, we compare the performance of the proposed algorithm with other clustering methods on text datasets. The performance of different methods is illustrated in Table 8. The table provides the *ACC* and *NMI* rates from top to bottom for each method, respectively. The results pertaining to other methods have been mentioned directly from the relevant publications.

The results indicate the higher performance of EDCWRN compared to other successful methods in this field. After EDCWRN, ADEC, and DCN methods have relatively good performance, respectively. As shown in Table 8, some state-of-the-art methods, such as SDEC, yield produce poor performance on the unbalanced *20NG* dataset. Compared to other algorithms, our model gets higher performance on all text datasets. Undoubtedly, EDCWRN is efficient in learning discriminative features and the samples are gradually well separated with the training procedure.

Comparing Tables 4 and 6 shows that most DEC-based methods have almost higher performance on image datasets than text datasets. This is because the original DEC method is highly efficient on balance datasets. As a result, improved DEC-based methods also work well for balance datasets. In the experiments, image datasets are balanced, while text datasets are unbalanced. The efficiency of DEC-based methods is significantly reduced for unbalanced text datasets. The applied representation weighting technique in the proposed method solves this problem. In the *20NG* dataset, which is highly unbalanced, the *ACC* and NMI have been significantly improved.

6 Conclusion

Many studies have been conducted to design an efficient deep clustering framework. These investigations have developed a suitable architecture or introduced a proper clustering loss function. In these methods, it was assumed that all the generated representations were of equal importance during the clustering procedure. However, all generated representations may not be suitable for clustering. Also, these methods used the

Table 7 The average performance of EDCWRN as well as other state-of-the-art methods

	DEC	VaDE	JULE	DEPICT	IDEC	DSCDAN	ADEC	DynAE	DeepCluster	LMVSC	EDCWRN
ACC	74.97	59.40	85.95	82.15	75.42	87.22	88.45	88.65	68.87	57.90	**88.70**
NMI	74.42	57.62	83.67	78.77	75.27	84.72	88.20	87.92	60.60	53.97	**88.27**

The best results are in boldfaced

Kullback–Leibler Divergence (KLD) loss function to train the model. This loss function does not preserve global data structure; therefore, it is generally agreed that clustering by the KLD loss function is not a good idea.

In this study, motivated by these weaknesses, we presented an efficient deep model to learn representations and cluster labels jointly, termed EDCWRN, applying an automatic local representation weighting strategy to weight the representations of each cluster properly. Moreover, we used the samples and their neighbors in the clustering procure to generate cluster-oriented meaningful representation. The formulation of the proposed algorithm was different from the other deep clustering methods, which significantly improved the results. In this study, we found that weighting the extracted representations has the most positive effect on the results and leads to better clustering. We also found that using the data neighbor technique to learn better representations is very effective. However, how neighbors are calculated can sometimes reduces network performance. Experimental results concerning four benchmark datasets indicated that the proposed algorithm generates better representations for clustering.

In future work, we would like to try other techniques for computing the feature weights. Moreover, it would be interesting to investigate the application of the EDCWRN in large image/text datasets. Estimating auxiliary target distribution (pseudo-labels) is an important step for deep clustering. Accurate estimation of these pseudo-labels can greatly contribute to optimal clustering. Studying other possible target distribution formulations can be an interesting line of research.

Table 8 The performances of the proposed algorithm and other clustering methods on text datasets

Method	Dataset	
	Reuters-10 k	*20NG*
DEC [25]	72.1	50.1
	49.7	45.3
DCN [52]	80.0	44.0
	76.0	48.0
PSSC [53]	78.0	–
	46.3	–
VaDE [42]	79.3	–
	52.1	–
SDEC [49]	67.9	31.04
	50.8	31.55
IDEC [22]	75.6	53.6
	49.8	44.4
DKM [48]	58.3	51.2
	33.1	46.7
DynAE [23]	73.2	47.5
	43.2	45.2
ADEC [30]	82.1	–
	60.5	–
EDCWRN	**83.7**	**58.1**
	60.6	**51.8**

The best results are in boldfaced

Appendix 1

Eq. (12) is optimized using backpropagation and SGD (mini-batch stochastic gradient descent.

Theorem 1 Let target distribution P be fixed, the gradients of $\mathcal{L}_c$ concerning embedded point $f_\theta(x_{im})$ and cluster center r_{jm} can be computed via Eq. (A.1):

Proof

$$\frac{\partial \mathcal{L}_c}{\partial f_\theta(x_{im})} = 2ab\sum_j \frac{p_{ij}\, w_{jm}\left(r_{jm}-x_{im}\right)\left(w_{jm}\left(r_{jm}-x_{im}\right)^2\right)^{b-1}}{a\left(w_{jm}\left(r_{jm}-x_{im}\right)^2\right)^b+1} \tag{A.1}$$

$$\frac{\partial \mathcal{L}_c}{\partial r_{jm}} = -2ab\sum_i \frac{p_{ij}\, w_{jm}\left(r_{jm}-x_{im}\right)\left(w_{jm}\left(r_{jm}-x_{im}\right)^2\right)^{b-1}}{a\left(w_{jm}\left(r_{jm}-x_{im}\right)^2\right)^b+1} \tag{A.2}$$

Then given a mini-batch with N samples and learning rate γ, r_{jm}, W (decoder's weights), W (encoder's weights), and are updated by Eqs. (A.3) to (A.5), respectively:

$$r_{jm} = r_{jm} - \frac{\gamma}{N}\sum_{n=1}^{N}\frac{\partial \mathcal{L}_c}{\partial r_{jm}} \tag{A.3}$$

$$W = W - \frac{\gamma}{N}\sum_{n=1}^{N}\frac{\partial \mathcal{L}_c}{\partial W} \tag{A.4}$$

$$W = W - \frac{\gamma}{N} \sum_{n=1}^{N} \left(\frac{\partial \mathcal{L}_r}{\partial W} + \lambda \frac{\partial \mathcal{L}_c}{\partial W} \right) \tag{A.5}$$

References

1. Oskouei AG, Hashemzadeh M, Asheghi B, Balafar M-A (2021) "CGFFCM: Cluster-weight and Group-local Feature-weight learning in Fuzzy C-Means clustering algorithm for color image segmentation," Appl Soft Comput, p. 108005, 2021/10/26/ 2021. https://doi.org/10.1016/j.asoc.2021.108005
2. Golzari Oskouei A, Balafar MA, Motamed C (2021) FKMAWCW: Categorical fuzzy k-modes clustering with automated attribute-weight and cluster-weight learning. Chaos, Solitons Fractals 153: 111494. https://doi.org/10.1016/j.chaos.2021.111494
3. Hashemzadeh M, Golzari Oskouei A, Farajzadeh N (2019) New fuzzy C-means clustering method based on feature-weight and cluster-weight learning. Appl Soft Comput 78:324–345. https://doi.org/10.1016/j.asoc.2019.02.038
4. Ke G, Hong Z, Yu W, Zhang X, Liu Z (2022) "Efficient multi-view clustering networks," Appl Intell https://doi.org/10.1007/s10489-021-03129-0
5. Pourbahrami S, Balafar MA, Khanli LM, Kakarash ZA (2020) A survey of neighborhood construction algorithms for clustering and classifying data points. Comp Sci Rev 38:100315. https://doi.org/10.1016/j.cosrev.2020.100315
6. Berahmand K, Nasiri E, Mohammadiani RP, Li Y (2021) Spectral clustering on protein-protein interaction networks via constructing affinity matrix using attributed graph embedding. Comput Biol Med 138:104933. https://doi.org/10.1016/j.compbiomed.2021.104933
7. Roshanzamir M, Balafar MA, Razavi SN (2017) Empowering particle swarm optimization algorithm using multi agents' capability: A holonic approach. Knowl-Based Syst 136:58–74. https://doi.org/10.1016/j.knosys.2017.08.023
8. Berahmand K, Bouyer A, Vasighi M (2018) Community detection in complex networks by detecting and expanding Core nodes through extended local similarity of nodes. IEEE Transactions on Computational Social Systems 5(4):1021–1033. https://doi.org/10.1109/TCSS.2018.2879494
9. Golzari Oskouei A, Hashemzadeh M (2022) CGFFCM: A color image segmentation method based on cluster-weight and feature-weight learning. Softw Impacts 11:100228. https://doi.org/10.1016/j.simpa.2022.100228
10. Li Y, Liao H (2021) Multi-view clustering via adversarial view embedding and adaptive view fusion. Appl Intell 51(3):1201–1212. https://doi.org/10.1007/s10489-020-01864-4
11. Bezdek JC (1981) Objective function clustering, in *Pattern recognition with fuzzy objective function algorithms*: Springer, pp 43–93. https://doi.org/10.1007/978-1-4757-0450-1_3
12. MacQueen J (1967) "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14: Oakland, CA, USA, pp 281–297
13. Huang X, Yang X, Zhao J, Xiong L, Ye Y (2018) A new weighting k-means type clustering framework with an l2-norm regularization. Knowl-Based Syst 151:165–179. https://doi.org/10.1016/j.knosys.2018.03.028
14. Zhao K, Dai Y, Jia Z, Ji Y (2021) General fuzzy C-means clustering algorithm using Minkowski metric. Signal Process 188:108161. https://doi.org/10.1016/j.sigpro.2021.108161
15. Wold S, Esbensen K, Geladi P (1987) Principal component analysis. Chemometr Intell Lab Syst 2(1):37–52. https://doi.org/10.1016/0169-7439(87)80084-9
16. Du C, Tian L, Chen B, Zhang L, Chen W, Liu H (2021) Region-factorized recurrent attentional network with deep clustering for radar HRRP target recognition. Signal Proc 183:108010. https://doi.org/10.1016/j.sigpro.2021.108010
17. Huang S, Kang Z, Xu Z, Liu Q (2021) Robust deep k-means: An effective and simple method for data clustering. Pattern Recognit 117:107996. https://doi.org/10.1016/j.patcog.2021.107996
18. Khan Z, Yang J (2020) Bottom-up unsupervised image segmentation using FC-Dense u-net based deep representation clustering and multidimensional feature fusion based region merging. Image Vision Comput 94:103871. https://doi.org/10.1016/j.imavis.2020.103871
19. Zhou Q, Zhou WA, Wang S (2021) Cluster adaptation networks for unsupervised domain adaptation. Image Vis Comput 108:104137. https://doi.org/10.1016/j.imavis.2021.104137
20. Xie M, Ye Z, Pan G, Liu X (2021) Incomplete multi-view subspace clustering with adaptive instance-sample mapping and deep feature fusion. Appl Intell 51(8):5584–5597
21. Das D, Ghosh R, Bhowmick B (2019) "Deep Representation Learning Characterized by Inter-Class Separation for Image Clustering," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 7–11 Jan. 2019, pp. 628–637. https://doi.org/10.1109/WACV.2019.00072
22. Guo X, Gao L, Liu X, Yin J (2017) "Improved deep embedded clustering with local structure preservation," in Ijcai, pp. 1753-1759
23. Mrabah N, Khan NM, Ksantini R, Lachiri Z (2020) Deep clustering with a Dynamic Autoencoder: From reconstruction towards centroids construction. Neural Netw 130:206–228. https://doi.org/10.1016/j.neunet.2020.07.005
24. Diallo B, Hu J, Li T, Khan GA, Liang X, Zhao Y (2021) Deep embedding clustering based on contractive autoencoder. Neurocomputing 433:96–107. https://doi.org/10.1016/j.neucom.2020.12.094
25. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis, International Conference on Machine Learning, Proceedings of Machine Learning Research 48:478–487. https://proceedings.mlr.press/v48/xieb16.html
26. Guo X, Zhu E, Liu X, Yin J (2018) Deep embedded clustering with data augmentation, in the Proceedings of The 10th Asian Conference on Machine Learning, Proceedings of Machine Learning Research 95:550–565. https://proceedings.mlr.press/v95/guo18b.html
27. Li F, Qiao H, Zhang B (2018) Discriminatively boosted image clustering with fully convolutional auto-encoders. Pattern Recogn 83:161–173
28. Chen D, Lv J, Zhang Y (2017) Unsupervised multi-manifold clustering by learning deep representation, in workshops at the thirty-first AAAI conference on artificial intelligence. https://aaai.org/ocs/index.php/WS/AAAIW17/paper/view/15099
29. Lu H, Chen C, Wei H, Ma Z, Jiang K, Wang Y (2022) Improved deep convolutional embedded clustering with re-selectable sample training. Pattern Recogn 127:108611. https://doi.org/10.1016/j.patcog.2022.108611
30. Mrabah N, Bouguessa M, Ksantini R (2020) Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift. IEEE Trans Knowl Data Eng
31. Fogel S, Averbuch-Elor H, Cohen-Or D, Goldberger J (2019) Clustering-driven deep embedding with pairwise constraints. IEEE Comput Graph Appl 39(4):16–27. https://doi.org/10.1109/MCG.2018.2881524
32. Balafar MA, Hazratgholizadeh R, Derakhshi MRF (2020) Active Learning for Constrained Document Clustering with Uncertainty

Region. Complexity 2020:3207306. https://doi.org/10.1155/2020/3207306
33. Xu C, Lin R, Cai J, Wang S (2022) Deep image clustering by fusing contrastive learning and neighbor relation mining. Knowl-Based Syst 238:107967. https://doi.org/10.1016/j.knosys.2021.107967
34. Wu L, Yuan L, Zhao G, Lin H, Li SZ (2022) Deep clustering and visualization for end-to-end high-dimensional data analysis. IEEE Transactions on Neural Networks and Learning Systems PP:1–12. https://doi.org/10.1109/TNNLS.2022.3151498
35. Xia H, Shao S, Hu C, Zhang R, Qiu T, Xiao F (2022) Robust clustering model based on attention mechanism and graph convolutional network. IEEE Trans Knowl Data Eng:1–1. https://doi.org/10.1109/TKDE.2022.3150300
36. Xia W, Wang Q, Gao Q, Zhang X, Gao X (2021) Self-supervised graph convolutional network for multi-view clustering. IEEE Transactions on Multimedia:1–1. https://doi.org/10.1109/TMM.2021.3094296
37. Qi C, Zhang J, Jia H, Mao Q, Wang L, Song H (2021) Deep face clustering using residual graph convolutional network. Knowl-Based Syst 211:106561. https://doi.org/10.1016/j.knosys.2020.106561
38. Berahmand K, Mohammadi M, Faroughi A, Mohammadiani RP (2022) A novel method of spectral clustering in attributed networks by constructing parameter-free affinity matrix. Clust Comput 25(2):869–888. https://doi.org/10.1007/s10586-021-03430-0
39. Berahmand K, Nasiri E, Rostami M, Forouzandeh S (2021) A modified DeepWalk method for link prediction in attributed social network. Computing 103(10):2227–2249. https://doi.org/10.1007/s00607-021-00982-2
40. McInnes L, Healy J, Saul N, Großberger L (2018) UMAP: Uniform Manifold Approximation and Projection, arXiv preprint arXiv:1802.03426. https://doi.org/10.48550/arxiv.1802.03426
41. Zhou P, Hou Y, Feng J (2018) Deep adversarial subspace clustering, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp 1596–1604
42. Jiang Z, Zheng Y, Tan H, Tang B, Zhou H (2017) "Variational deep embedding: an unsupervised and generative approach to clustering, " in *proceedings of the 26th international joint conference on artificial intelligence*, pp. 1965-1972
43. Yang J, Parikh D, Batra D (2016) Joint unsupervised learning of deep representations and image clusters, in Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR) pp 5147–5156
44. Guo X, Liu X, Zhu E, Yin J (2017) Deep clustering with convolutional autoencoders, in International conference on neural information processing, Springer 10635:373–382. https://doi.org/10.1007/978-3-319-70096-0_39
45. Caron M, Bojanowski P, Joulin A, Douze M (2018) "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149
46. Ghasedi Dizaji K, Herandi A, Deng C, Cai W, Huang H (2017) "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE international conference on computer vision*, pp. 5736–5745
47. Yang X, Deng C, Zheng F, Yan J, Liu W (2019) "Deep spectral clustering using dual autoencoder network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4066–4075
48. Moradi Fard M, Thonet T, Gaussier E (2020) Deep k-Means: Jointly clustering with k-Means and learning representations. Pattern Recogn Lett 138:185–192. https://doi.org/10.1016/j.patrec.2020.07.028
49. Ren Y, Hu K, Dai X, Pan L, Hoi SCH, Xu Z (2019) Semi-supervised deep embedded clustering. Neurocomputing 325:121–130. https://doi.org/10.1016/j.neucom.2018.10.016
50. Kang Z, Zhou W, Zhao Z, Shao J, Han M, Xu Z (2020) Large-scale multi-view subspace clustering in linear time. Proceedings of the AAAI Conference on Artificial Intelligence 34(04):4412–4419
51. Zhang T, Ji P, Harandi M, Hartley R, Reid I (2018) Scalable deep k-subspace clustering, in *Asian Conference on Computer Vision*: Springer pp 466–481. https://doi.org/10.1007/978-3-030-20873-8_30
52. Yang B, Fu X, Sidiropoulos ND, Hong M (2017) "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in the 34th International Conference on Machine Learning, Proceedings of Machine Learning Research 70:3861–3870. https://proceedings.mlr.press/v70/yang17b.html
53. Lv J, Kang Z, Lu X, Xu Z (2021) Pseudo-supervised deep subspace clustering. IEEE Trans Image Process 30:5252–5263. https://doi.org/10.1109/TIP.2021.3079800
54. Aria M, Nourani E, Golzari Oskouei A (2022) ADA-COVID: Adversarial Deep Domain Adaptation-Based Diagnosis of COVID-19 from Lung CT Scans Using Triplet Embeddings. Comput Intell Neurosci 2022:2564022. https://doi.org/10.1155/2022/2564022
55. H. Akramifard, M. Balafar, S. Razavi, and A. R. Ramli, "Emphasis Learning, Features Repetition in Width Instead of Length to Improve Classification Performance: Case Study—Alzheimer's Disease Diagnosis," Sensors, vol. 20, no. 3, p. 941, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/3/941

Amin Golzari Oskouei received the B.Eng. degree in Information Technology from the Payame Noor University, Iran, in 2016, and the Master degree in Information Technology from the Azarbaijan Shahid Madani University, Tabriz, Iran, in 2018. He was a member of the Artificial Intelligence and Machine Learning research Laboratory of the Faculty of Information Technology and Computer Engineering of Azarbaijan Shahid Madani University, from 2016 until 2019. His research interests include Artificial Intelligence, Machine Learning, Deep Learning, and Data Mining.

Mohammad Ali Balafar is currently Professor in the Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz, Iran, and the director of Information Technology and Intelligent Multimedia Research Lab. His research interests include Artificial Intelligence, Machine Learning, Pattern Recognition, and Computer Vision.

Cina Motamed is Professor in Computer Science in the University of Orléans, France. He received his B.Sc. in mathematics, and M.Sc in Electrical Engineering and Computer Science from the University of Caen, France, and the Ph.D. degree in Computer Science from the University of Compiegne, France, in 1987, 1989, and 1992, respectively. Its research is focused on the automatic visual surveillance using computer Visionand Pattern Recognition techniques. The main research interests focus on the design of the multi-camera system for real-time multi-object tracking and human action recognition. He works on the uncertainty management over the vision system by using graphical models, and beliefs propagation. He is currently interested by unsupervised learning approaches and deep Learning Techniques in pattern recognition applications.