

Deep Clustering with Self-Supervision using Pairwise Data Similarities

Mohammadreza Sadeghi^{1,2*} and Narges Armanfard^{1,2}

^{1*}Department of Electrical and Computer Engineering, McGill University, 3380 Blvd Robert-Bourassa, Montreal, H2X 2G6, QC, Canada.

²Mila - Quebec AI Institute, 6666 Rue Saint-Urbain, Montreal, H2S 3H1, QC, Canada.

*Corresponding author(s). E-mail(s): mohammadreza.sadeghi@mcgill.ca;
Contributing authors: narges.armanfard@mcgill.ca;

Abstract

Deep clustering incorporates embedding into clustering to find a lower-dimensional space appropriate for clustering. In this paper, we propose a novel deep clustering framework with self-supervision using pairwise data similarities (DCSS). The proposed method consists of two successive phases. In the first phase, we propose to form hypersphere-like groups of similar data points, i.e. one hypersphere per cluster, employing an autoencoder that is trained using cluster-specific losses. The hyper-spheres are formed in the autoencoder’s latent space. In the second phase, we propose to employ pairwise data similarities to create a \mathbf{K} -dimensional space that is capable of accommodating more complex cluster distributions, hence providing more accurate clustering performance. \mathbf{K} is the number of clusters. The autoencoder’s latent space obtained in the first phase is used as the input of the second phase. The effectiveness of both phases is demonstrated on seven benchmark datasets by conducting a rigorous set of experiments. The DCSS code is available: <https://github.com/Armanfard-Lab/DCSS>

Keywords: Deep clustering, autoencoder, pairwise data similarity, clustering with soft assignments, cluster-specific loss.

1 Introduction

Many science and practical applications, information about category (aka label) of data samples is non-accessible or expensive to collect. Clustering, as a major data

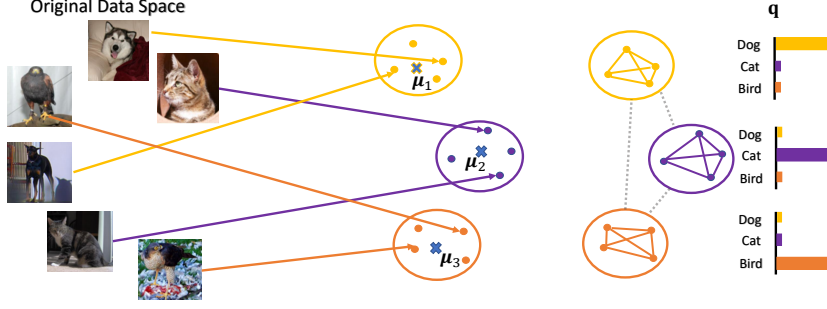


Fig. 1 The motivation of the proposed DCSS method. Arrows show the nonlinear mapping, using the AE, from the original input space to the AE’s latent space (i.e. the u space). DCSS employs pairs of similar and dissimilar samples to create the K -dimensional space q in which pairwise similarities and dissimilarities are strengthened. Similar samples are connected with solid lines, and dashed lines represent dissimilar data.

analysis tool in pattern recognition and machine learning, endeavors to gather essential information from unlabeled data samples. The main goal of clustering methods is to partition data points based on a similarity metric.

Deep learning-based clustering methods have been widely studied, and their effectiveness is demonstrated in many applications such as image segmentation [1], social network analysis [2], face recognition [3, 4], and machine vision [5, 6]. The common practice in these methods is to map the original feature space onto a lower dimensional space (aka latent space) in which similar samples build data groups that can be detected by a simple method like k-means [7].

One of the most common approaches in obtaining the lower dimensional space is based on autoencoder (AE) and its variations [8, 9, 10, 11, 12]. An AE consists of two networks: an encoder and a decoder. The encoder maps the original input space onto a latent space while the decoder tries to reconstruct the original space using the encoder’s output space. Encoder and decoder networks are trained to minimize a loss function that contains the data reconstruction losses. The AE’s latent space, whose dimension is much lower than the dimension of the original input space, is indeed a nonlinear transformation of the original space.

Some more advanced AE-based clustering methods, e.g. [13, 14, 15, 16, 17], include in their loss function the data clustering losses besides the reconstruction losses. This makes the AE’s latent space more effective for data clustering. Despite the reconstruction loss that can be directly computed based on the difference between the encoder’s input and the decoder’s output, calculating the *true* clustering loss is impossible due to the unsupervised nature of the clustering problem where the true cluster label of the data points remains unknown during the training phase. Hence, researchers employ an *approximation* of the clustering losses when training the networks. At each training iteration, the approximation is calculated based on the data distribution in the latent space obtained in the previous iteration.

Many existing algorithms in the field, e.g. [13, 14, 18, 19, 20, 21], approximate the data clustering losses by first performing a crisp cluster assignment and then calculating the clustering losses using the crisply clustered data, based on a criterion such as

the level of compactness or density of the data within clusters. Crisp assignment (as opposed to soft assignment) assigns a data point to only one cluster – e.g., to the one with the closest center. However, such crisp cluster assignment of the data in an intermediate training iteration may mislead the training procedure if a significant number of samples are mis-clustered, as the error propagates to the following iterations. This crisp assignment issue would be more serious if there exists a high uncertainty when deciding to which cluster a data point should be assigned – e.g., the extreme uncertainty would be related to the case where a data point is equally close to all cluster centers. Recently, self-supervised learning [22, 23, 24, 25, 26] has garnered significant attention within the domain of unsupervised representation learning and clustering. For instance, the Simple Framework for Contrastive Learning for Visual Representation Learning (SimCLR) [22] and Contrastive Clustering (CC) [23] have departed from the conventional practice of relying on pseudo labels for training neural networks. Instead, these methods generate two augmentations of a given data sample and treat the two augmentations of the same sample as positive pairs while designating all other samples in the batch as negative pairs, even if they correspond to the same object in different images. This approach, which treats the same objects in two different images as negative pairs, may inadvertently impact the network’s ability to converge towards an optimal solution for the problem.

To the best of our knowledge, only a limited number of deep clustering algorithms, such as [27, 28, 29, 30], attempt to utilize the pairwise relationships between sample pairs in an unsupervised manner. However, it should be noted that some of these methods, including [27, 29], encounter challenges related to the presence of a high number of false positive pairs during their model training, primarily because they design their loss functions to include more and more samples till it includes all samples rather than focusing on highly confident samples. All other deep clustering algorithms neglect the important relevant information available in sample pairs while the effectiveness of such information has been proven in the *supervised* and *semi-supervised* learning methods [31, 32, 33, 34, 35] where the data class labels are employed during the training phase. For example, metric learning algorithms are supervised learning techniques that learn a distance metric employing pairwise distances [31, 32]. Their goal is to decrease the distance between similar samples, i.e. samples from the same class, and increase the distance between dissimilar samples, i.e. samples from different classes. Contrastive learning is a metric learning technique that aims to maximize the similarities of positive pairs while minimizing those of negative pairs where the positive and negative pairs, respectively, refer to the samples from the same and different classes.

Furthermore, to the best of our knowledge, the existing clustering methods utilize a single loss function for all data clusters, ignoring the possible existence of differences between the characteristics of the different clusters. We present novel cluster-specific losses in this paper, designed to not only emphasize the reconstruction and centering of data samples around cluster centers but also to aid the AE in finding distinctive properties within each cluster. The only existing algorithm that explicitly employs cluster-specific losses is presented in [18]. However, this algorithm suffers from high computational cost as the algorithm requires training of K distinct AEs, where K is the number of data clusters.

In summary, to the best of our knowledge, all existing AE-based unsupervised clustering methods suffer from at least one of the following: the crisp assignment issue, ignoring the relevant, useful information available in the data pairs, failure to reliably identify similar and dissimilar samples, treating all clusters similarly through minimizing a single common loss function for all data clusters.

In this paper, we propose a novel AE-based clustering algorithm called Deep Clustering with Self-Supervision, DCSS, that addresses all the aforementioned drawbacks. DCSS is a novel unified framework that employs pairwise data similarities as a means of self-supervision during its training procedure. DCSS mitigates the error propagation issue caused by the uncertain crisp assignments by employing the soft assignments in the loss function. DCSS considers an individual loss for every data cluster, where a loss consists of weighted reconstruction and clustering errors. A sample’s clustering error is calculated using the sample’s Euclidean distance to the cluster centers. This results in obtaining a latent space (for the AE), called \mathbf{u} space, in which similar data points form hypersphere-like clusters, one hypersphere per cluster. To make the cluster distributions more distinguishable from each other and to accommodate more complex distributions, we propose to employ pairwise data similarities and train a K -dimensional space \mathbf{q} in which a pair of similar (dissimilar) samples sit very close to (far from) each other, where K is the number of data clusters and inner product is used for similarity measurement. We define \mathbf{q} to be the last layer of a fully connected network called MNet. Only similar and dissimilar pairs of samples contribute to training MNet. Due to the curse of dimensionality [36], similar and dissimilar samples are not recognizable in the original input feature space. Instead, we propose to measure the pairwise similarities in the partially trained \mathbf{u} and \mathbf{q} spaces, which are more reliable¹ for similarity measurement. The input layer of MNet is the output layer of the AE’s encoder network, i.e., the \mathbf{u} space. Our experiments, supported by mathematical proofs, demonstrate that the data representations in the \mathbf{q} space are very close to one-hot vectors where the index of the most active element points the true cluster label out. Furthermore, we demonstrate that the DCSS method can be employed as a general framework to improve the performance of the existing AE-based clustering methods, e.g. [37, 15, 14, 13]. An intuitive motivation of the proposed method is illustrated in Fig. 1.

The rest of this paper is organized as follows. Section 2 presents a brief review of deep clustering methods. Section 3 presents details of the proposed DCSS framework. Extensive experimental results that demonstrate the effectiveness of the DCSS method are presented in Section 4. Section 5 conveys the gist of this paper.

2 Related Work

An exhaustive review of previous works is beyond the scope of this paper. We refer to the survey of Xu et al. [38] on non-deeplearning-based clustering methods. The following focuses on the review of some related deep clustering methods.

¹In this paper, space A is considered more reliable than space B if and only if the clustering performance in A is better than in B.

2.1 Deep learning based methods

Deep neural networks (DNN) have been widely used to tackle the unsupervised clustering problem. These algorithms try to train a DNN-based model in an unsupervised manner [39, 40, 30, 41, 42]. In [30], the authors initially identify the N nearest neighbors for each data sample in the latent representation of a network. This network is initialized to solve a pretext task. Subsequently, they minimize a loss function based on the pairwise relationship between nearest neighbor samples for all samples in the dataset. It is important to highlight that, as they formulate their loss function based on the entire dataset, in the event of a poor initialization leading to their method being trapped in a local minimum, there is no opportunity for escape. Additionally, including all samples in the loss function and identifying N nearest neighbors for each sample results in a high computational cost for this algorithm. [43] encourages predicted representations of the augmented data points to be close to those of the original data points by maximizing the information-theoretic dependency between data and their predicted representations. RUC [44] proposes a two-step method where, in the first step, it endeavors to clean the dataset, and in the second step, it retrains the network with the purified dataset. [45] trains a DNN in an unsupervised manner to extract an indicator feature vector for each data sample. It then uses the obtained vectors to assign the data points to different clusters. Recently, contrastive learning has attracted researchers' attention in the unsupervised clustering field [23, 22, 46]. As is discussed in Section 1, such algorithms first need to construct negative and positive pairs by applying augmentation to the data points. They then map the data into a feature space and endeavor to maximize similarity (minimize dissimilarity) in positive (negative) pairs. An extensive review of the DNN-based methods can be found in [47].

Among the DNN-based models, the AE-based and generative-based algorithms have been widely studied and used for unsupervised data clustering. These two categories are reviewed in the following sections.

2.1.1 AE-based algorithms

AE-based algorithms utilize deep autoencoders to embed original data points in a lower-dimensional space. In some algorithms, such as [48, 49], learning the lower representation of the data points is separated from the clustering task. In [48], an AE is used to find a lower-dimensional representation of data points by enforcing group sparsity and locality-preserving constraints. The cluster assignments are then obtained by applying the k-means algorithm to the obtained lower-dimensional space. Graph clustering [50, 51] is a key branch of clustering that tries to find disjoint partitions of graph nodes such that the connections between nodes within the same partition are much denser than those across different partitions. [49] takes advantage of a deep autoencoder to find a lower-dimensional representation of a graph; it then utilizes the k-means algorithm to define clusters in the lower-dimensional space.

In order to further improve clustering performance, more recent AE-based algorithms simultaneously embed data points in a lower-dimensional feature space and perform clustering using the obtained space. Deep embedded clustering (DEC) [37] first trains a stacked autoencoder layer by layer using the reconstruction losses and

then removes the decoder and updates the encoder part by minimizing a Kullback–Leibler (KL) divergence between the distribution of soft assignments and a pre-determined target distribution. Soft assignments are the similarity between data points and cluster centers and are calculated using Students’ t-distribution. Due to the unsupervised nature of the clustering problem, the target distribution of the data points is unknown. Hence, DEC uses an arbitrary target distribution, which is based on the squared of the soft assignments. Despite the DEC method, a few recent studies, e.g [14, 13, 15], propose to take advantage of the AE’s decoder as well as the encoder. These algorithms use notions of both reconstruction and clustering losses with the goal of maintaining the local structure of the original data points while training the algorithm’s networks. For example, improved deep embedding clustering (IDEC) [15] tries to improve the clustering performance of DEC by considering the reconstruction loss of an AE besides the KL divergence loss of DEC. Improved deep embedding clustering with fuzzy supervision (IDECF) [17] improves the DEC method by employing both reconstruction and clustering losses and estimating the target distribution through training a deep fuzzy c-means network. Deep clustering network (DCN) [14] jointly learns a lower-dimensional representation and performs clustering. DCN trains its AE by minimizing a combination of the reconstruction loss and the objective function of the k-means algorithm. This results in a k-means-friendly latent space. DCN updates AE’s parameters and cluster centers separately. The latter is based on solving a discrete optimization problem. In the deep k-means (DKM) algorithm [13], which has the same objective function as DCN, both network parameters and cluster centers are updated simultaneously by minimizing its objective function using stochastic gradient descent.

Spectral clustering [52, 53] is a clustering approach that is based on building a graph of data points in the original space and then embedding the graph into a lower-dimensional space in which similar samples sit close to each other. Spectral clustering has been employed in DNN-based methods [16, 54]. For example, deep spectral clustering (DSC) is recently presented in [16]. DSC has a joint learning framework that creates a low-dimensional space using a dual autoencoder that has a common encoder network and two decoder networks. The first decoder tries to reconstruct the original input from the AE’s latent space, and the second decoder endeavors to denoise the encoder’s latent space. DSC considers reconstruction, mutual information, and spectral clustering losses for networks’ training.

2.1.2 Deep generative based algorithms

Variational autoencoders (VAEs) [55] and Generative adversarial networks (GANs) [56] are among the most well-known deep generative models which are effective for data clustering. For example, variational deep embedding (VaDE) [57] finds a latent space that captures the data statistical structure that can be used to produce new samples. The data generative process in VaDE is based on a Gaussian Mixture Model (GMM) and a deep neural network. Deep adversarial clustering [58] is another generative model that applies the adversarial autoencoder [59] to clustering. The adversarial autoencoder employs an adversarial training procedure to match the aggregated posterior of the latent representation with a Gaussian Mixture distribution. [58] objective

function includes a reconstruction term, Gaussian mixture model likelihood, and the adversarial objective. More generative-based models can be found in [47]. GAN is a method of training a generative model by framing the problem as a supervised learning task with two sub-models: the generator model that is trained to generate new samples and the discriminator model that tries to classify examples as either real or fake (generated). Many GAN-based algorithms have been developed for clustering tasks [47, 60]. [61] presents a GAN-based algorithm that learns disentangled representations in an unsupervised manner. It maximizes the mutual information between a small subset of the latent variables and the observation. [62] expands the idea of GMM to the GAN mixture model (GANMM) by devising a GAN model for each cluster. GAN-based clustering algorithms suffer from vanishing gradients and mode collapse.

3 Proposed Method

Consider a K-clustering problem that aims to partition a given dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ into K disjoint clusters, where \mathbf{x}_i indicates the i th data sample, N is the number of data points, and K is a predefined user-settable parameter. DCSS utilizes an AE consisting of an encoder and a decoder network, respectively denoted by $f(\cdot)$ and $g(\cdot)$. Latent representation of X is denoted by $U = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N\}$, where $\mathbf{u}_i = f(\mathbf{x}_i; \boldsymbol{\theta}_e) \in \mathbb{R}^d$, d indicates dimension of the latent space, and $\boldsymbol{\theta}_e$ denotes parameters of the encoder network. The reconstructed output of the AE is denoted by $\hat{\mathbf{x}}_i = g(\mathbf{u}_i; \boldsymbol{\theta}_d)$, where $\boldsymbol{\theta}_d$ represents the decoder parameters. The center of the k th data group in the \mathbf{u} space is denoted by $\boldsymbol{\mu}^{(k)}$. To accommodate complex cluster distributions, we propose to employ pairwise data similarities in DCSS. To this end, we employ the fully connected network MNet which takes the latent representation of each data point, i.e. \mathbf{u}_i , as input and maps it to a K -dimensional vector \mathbf{q}_i which its k th element indicates the probability of \mathbf{x}_i belonging to the k th data cluster. In this paper, the output of MNet for the i th data point is denoted by $\mathbf{q}_i = M(\mathbf{u}_i; \boldsymbol{\theta}_M)$, where $M(\cdot)$ and $\boldsymbol{\theta}_M$ respectively shows MNet and its corresponding parameters.

The proposed DCSS method consists of two phases. The first phase is to provide hypersphere-like data clusters through training an AE using weighted reconstruction and centering losses, and the second phase is to employ pairwise data similarities to self-supervise the remaining training procedure.

3.1 Phase 1: AE training

At each training batch \mathfrak{B} , we propose to train the AE in K successive runs, where at each run, a specific loss corresponding to a specific data cluster is minimized. More specifically, at the k th run, the AE focuses on the reconstruction and centering of the data points that are more probable to belong to the k th data cluster.

The loss function of the k th run, i.e. $\mathcal{L}_u^{(k)}$, is shown in (1a) where $\mathcal{L}_r^{(k)}$ and $\mathcal{L}_c^{(k)}$, shown in (1b) and (1c), respectively denotes weighted summation of the sample reconstruction and centering losses. α is a hyperparameter indicating the importance of centering loss vs. reconstruction loss. m indicates the level of fuzziness and is set to

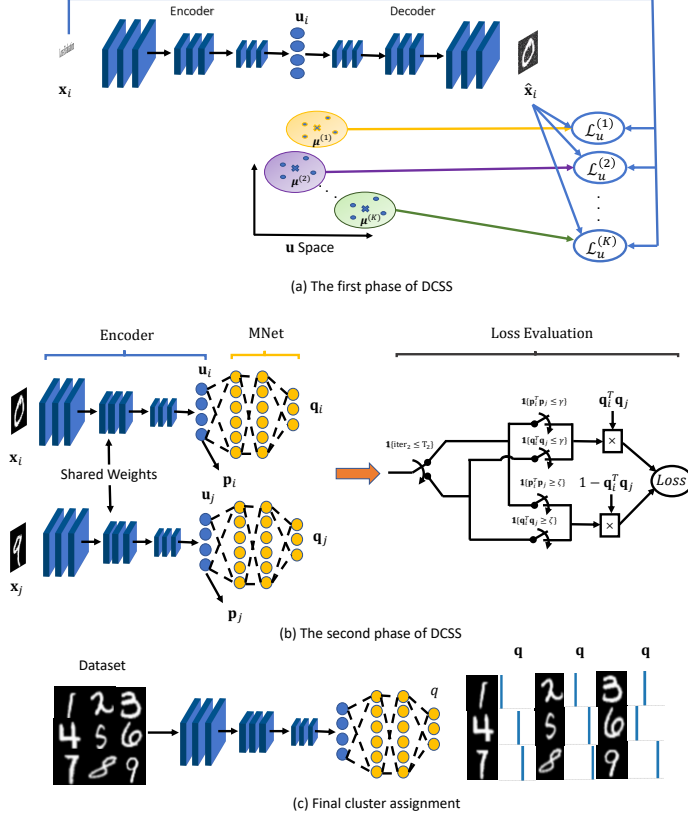


Fig. 2 (a) Training scheme of the first phase of DCSS. (b) Training procedure of the second phase of DCSS; at the outset, when $\text{iter}_2 \leq T_2$, MNet is trained based on the pairwise similarities defined in the \mathbf{u} space – i.e. The similarity between two data points \mathbf{x}_i and \mathbf{x}_j is determined using the dot product of \mathbf{p}_i and \mathbf{p}_j . At the later stages of MNet training, when $\text{iter}_2 > T_2$, the pairwise similarities are measured in the \mathbf{q} space itself using $\mathbf{q}_i^T \mathbf{q}_j$. (c) Visualization of the final cluster assignment using DCSS; after completing the training phases shown in (a) and (b), we cluster a data point by locating the largest element of its representation in the \mathbf{q} space.

1.5 in all experiments.

$$\mathcal{L}_u^{(k)} = \mathcal{L}_r^{(k)} + \alpha \mathcal{L}_c^{(k)} \quad (1a)$$

$$\mathcal{L}_r^{(k)} = \sum_{\mathbf{x}_i \in \mathfrak{B}} p_{ik}^m \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2 \quad (1b)$$

$$\mathcal{L}_c^{(k)} = \sum_{\mathbf{x}_i \in \mathfrak{B}} p_{ik}^m \|\mathbf{u}_i - \boldsymbol{\mu}^{(k)}\|_2^2 \quad (1c)$$

Where

$$p_{ik} = \frac{\frac{1}{\|\mathbf{u}_i - \boldsymbol{\mu}^{(k)}\|_2^{2/(m-1)}}}{\sum_{j=1}^K \frac{1}{\|\mathbf{u}_i - \boldsymbol{\mu}^{(j)}\|_2^{2/(m-1)}}} \quad (2)$$

Since data clustering is an unsupervised task, the data cluster memberships are unknown at the problem’s outset. As such, at the k th run, we use the Euclidean distance between \mathbf{u}_i and $\boldsymbol{\mu}^{(k)}$ as a means of measuring the membership degree of \mathbf{x}_i to the k th data cluster, denoted by p_{ik} defined in (2) where $\mathbf{p}_i = [p_{i1}, \dots, p_{iK}]$. The cluster memberships are used as the sample weights in (1b) and (1c). The closer a sample is to the cluster center $\boldsymbol{\mu}^{(k)}$, the higher contribution that sample has in minimizing the loss function corresponding to the k th run.

Every T_1 number of training epochs, we update the centers to the average of weighted samples in the \mathbf{u} space, as is shown in (3), where samples closer to $\boldsymbol{\mu}^{(k)}$ have more contribution to updating.

$$\boldsymbol{\mu}^{(k)} = \frac{\sum_{\mathbf{x}_i \in X} p_{ik}^m \mathbf{u}_i}{\sum_{\mathbf{x}_i \in X} p_{ik}^m} \quad (3)$$

The block diagram of the first phase is shown in Fig. 2(a). As is demonstrated in our experiments (see Section 4.5), minimizing (1a) results in forming hypersphere-like groups of similar samples in the \mathbf{u} space, one hypersphere per cluster.

3.2 Phase 2: self-supervision using pairwise similarities

To allow accommodating non-hypersphere shape distributions and to employ the important information available in the pairwise data relations, we propose to append a fully connected network, called MNet, to the encoder part of the AE, trained in Phase 1 while discarding its decoder network. The MNet’s output layer, i.e. the \mathbf{q} space, consists of K neurons where each neuron corresponds to a data cluster. We utilize the soft-max function at the output layer to obtain probability values employed for obtaining the final cluster assignments. More specifically, for an input sample \mathbf{x}_i , the output value at the j th neuron, i.e. q_{ij} , denotes the probability of \mathbf{x}_i belonging to the j th cluster.

MNet aims to strengthen (weaken) similarities of two similar (dissimilar) samples. MNet parameters, i.e., $\boldsymbol{\theta}_M$, are initialized with random values. Hence, at the first few training epochs, when \mathbf{q} is not yet a reliable space, pairwise similar and dissimilar samples are identified in the \mathbf{u} space. Then, after a few training epochs, pairs of similar and dissimilar samples are identified in the \mathbf{q} space. In both of the \mathbf{u} and \mathbf{q} spaces, we define two samples as similar (dissimilar) if the inner product of their corresponding cluster assignment vectors is greater (lower) than threshold ζ (γ). More specifically, knowing that the k th element of \mathbf{p}_i (\mathbf{q}_i) denotes the membership of \mathbf{x}_i to the k th cluster in the \mathbf{u} (\mathbf{q}) space, the inner product of \mathbf{p}_i (\mathbf{q}_i) and \mathbf{p}_j (\mathbf{q}_j) is considered as the notion of similarity between data points \mathbf{x}_i and \mathbf{x}_j .

The loss function proposed for the MNet training, at the first T_2 training epochs, is shown in (4) where ζ and γ are two user-settable hyperparameters, and $\mathbb{1}\{\cdot\}$ is the indicator function.

$$\mathcal{L}_M = \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathfrak{B}} \mathbb{1}\{\mathbf{p}_i^T \mathbf{p}_j \geq \zeta\}(1 - \mathbf{q}_i^T \mathbf{q}_j) + \mathbb{1}\{\mathbf{p}_i^T \mathbf{p}_j \leq \gamma\}(\mathbf{q}_i^T \mathbf{q}_j) \quad (4)$$

Algorithm 1 Clustering procedure using DCSS

Input: Data points X , θ_e , θ_d , θ_M , $\mu^{(k)}$ for $k = 1, \dots, K$

Output: θ_e , θ_M

Phase 1:

- 1: Initialize θ_e and θ_d with a pre-trained network (see Section 2 of the supplementary material).
- 2: **for** $\text{iter}_1 \in \{1, 2, \dots, \text{MaxIter}_1\}$ **do**
- 3: **for** $k \in \{1, 2, \dots, K\}$ **do**
- 4: Compute p_{ik} using (2), for $i \in \mathfrak{B}$
- 5: Update AE's parameters by employing (1a) as loss function
- 6: **end for**
- 7: Every T_1 iterations, update cluster centers using (3)
- 8: **end for**

Phase 2:

- 9: **for** $\text{iter}_2 \in \{1, 2, \dots, \text{MaxIter}_2\}$ **do**
- 10: **if** $\text{iter}_2 \leq T_2$ **then**
- 11: Compute vectors \mathbf{p}_i for $i \in \mathfrak{B}$
- 12: Compute vectors \mathbf{q}_i for $i \in \mathfrak{B}$
- 13: Update θ_e and θ_M to minimize (4)
- 14: Update centers $\mu^{(k)}$, $k = 1, \dots, K$, using (3)
- 15: **else**
- 16: Compute \mathbf{q}_i for $i \in \mathfrak{B}$
- 17: Update θ_e and θ_M to minimize (5)
- 18: **end if**
- 19: **end for**

Final Cluster Assignments:

- 20: Compute \mathbf{q}_i for \mathbf{x}_i , $i = 1, \dots, N$
 - 21: Assign each data sample to the most probable cluster
-

As can be inferred from (4), only similar and dissimilar samples, identified in the \mathbf{u} space, contribute to the MNet training and a pair of samples with a similarity value between ζ and γ , i.e. in the ambiguity region, does not contribute to the current training epoch. Therefore, minimizing \mathcal{L}_M strengthens (weakens) the similarity of similar (dissimilar) samples in the \mathbf{q} space. Along with training the MNet parameters, the encoder parameters θ_e are also updated through back-propagation in an end-to-end manner. After completing each training epoch, centers $\mu^{(k)}$, $k = 1, \dots, K$, are also updated using (3).

After T_2 epochs, when \mathbf{q} becomes a relatively reliable space for identifying similar and dissimilar samples, we further train MNet using the loss function \mathcal{L}'_M defined in (5). A pair contributes to \mathcal{L}'_M if its corresponding similarity value, in the \mathbf{q} space, is not in the ambiguity region. As is demonstrated in Section 4, as the MNet training phase progresses, more and more pairs contribute to the training procedure. Again, the \mathbf{u}

space receives small updates through the backpropagation process when minimizing \mathcal{L}'_M .

$$\mathcal{L}'_M = \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathfrak{B}} \mathbb{1}\{\mathbf{q}_i^T \mathbf{q}_j \geq \zeta\}(1 - \mathbf{q}_i^T \mathbf{q}_j) + \mathbb{1}\{\mathbf{q}_i^T \mathbf{q}_j \leq \gamma\}(\mathbf{q}_i^T \mathbf{q}_j) \quad (5)$$

Fig. 2(b) shows the overall training procedure of the DCSS's second phase.

3.3 Final cluster assignments

To determine the final cluster assignment of a data point \mathbf{x}_i , we utilize the trained encoder and MNet networks to obtain the data representation in the \mathbf{q} space, i.e. \mathbf{q}_i . \mathbf{x}_i is assigned to the most probable cluster, i.e. the index corresponding to the highest element of \mathbf{q}_i is the cluster label of \mathbf{x}_i . Such clustering assignment process is shown in Fig. 2(c).

The pseudo-code of the DCSS algorithm is presented in Algorithm 1.

3.4 Proper choice of ζ and γ

In this section, we discuss the selection of optimal hyperparameters ζ and γ for our model. It is crucial to choose appropriate values for these hyperparameters, as the model's performance is greatly impacted by their values. When the value of ζ is high and γ is small (e.g., $\zeta = 0.9$ and $\gamma = 0.1$), the model tends to consider only a small subset of the available data during the second training phase, resulting in the neglect of crucial information between true similar and dissimilar samples. Conversely, when the value of ζ is low and γ is high (e.g., $\zeta = 0.5$ and $\gamma = 0.5$), the model may struggle to accurately distinguish between similar and dissimilar samples. Therefore, selecting optimal values for ζ and γ is critical to ensure the effectiveness of our model.

Notation clarification: Representation of the i th sample in the \mathbf{q} space is shown by \mathbf{q}_i . The k th element of \mathbf{q}_i is shown by $q_{ik}, k = 1, \dots, K$, where K is the number of data clusters. Note that \mathbf{q}_i is the MNet output when the input sample is x_i . Since we employ soft-max as the final layer of MNet, $0 \leq q_{il} \leq 1$ where $1 \leq l \leq K$ and the ℓ_1 -norm of \mathbf{q}_i is equal to 1. Furthermore, as is discussed in the manuscript, parameters ζ and γ are values between 0 and 1.

Definition 3.1. Two data points, i.e. i and j , are adjacent (aka similar) if and only if $\mathbf{q}_i^T \mathbf{q}_j \geq \zeta$.

Definition 3.2. Two data points, i.e. i and j , are in the same cluster if and only if the index of the maximum value in their corresponding \mathbf{q} vector (i.e. \mathbf{q}_i and \mathbf{q}_j) are equal.

Theorem 1. Consider the i th and j th data points. Then :

$$\mathbf{q}_i^T \mathbf{q}_j \leq \min \left\{ \max_l \{q_{il}\}, \max_l \{q_{jl}\} \right\} \quad (6)$$

where $\mathbf{q}_i^T \mathbf{q}_j$ is the inner product of the two vector \mathbf{q}_i and \mathbf{q}_j .

Proof. Assume \mathbf{q}_j^* is a maximal vector that satisfies the below inequality:

$$\mathbf{q}_i^T \mathbf{q}_j \leq \mathbf{q}_i^T \mathbf{q}_j^*, \quad (7)$$

where $\|\mathbf{q}_j^*\|_1 = 1$. In addition, assume the index of the maximum element of \mathbf{q}_i is r :

$$r = \arg \max_l \{q_{il}\} \quad (8)$$

In the following, we first prove by contradiction that \mathbf{q}_j^* must be a one-hot vector. Then we prove (6).

Assume \mathbf{q}_j^* is not a one-hot vector. Therefore, there exists at least one index, i.e. e , that its corresponding element q_{je}^* is non-zero:

$$\exists e : e \neq r \text{ and } q_{je}^* \neq 0. \quad (9)$$

Now, let's define a vector \hat{q}_j as follows:

$$\hat{q}_{jl} = \begin{cases} q_{je}^* + q_{jr}^* & \text{if } l = r \\ 0 & \text{if } l = e \\ q_{jl}^* & \text{O.W} \end{cases} \quad (10)$$

where \hat{q}_{jl} denotes the l^{th} element of $\hat{\mathbf{q}}_j$. Since $\|\mathbf{q}_j^*\|_1 = 1$, we can immediately show that $\|\hat{\mathbf{q}}_j\|_1 = 1$. Moreover, we can represent the inner products $\mathbf{q}_i^T \mathbf{q}_j^*$ and $\mathbf{q}_i^T \hat{\mathbf{q}}_j$ as shown in (11) and (12), respectively.

$$\mathbf{q}_i^T \mathbf{q}_j^* = q_{ir} q_{jr}^* + q_{ie} q_{je}^* + \sum_{l \neq r, e} q_{il} q_{jl}^* \quad (11)$$

$$\mathbf{q}_i^T \hat{\mathbf{q}}_j = q_{ir} (q_{jr}^* + q_{je}^*) + q_{ie} \times 0 + \sum_{l \neq r, e} q_{il} q_{jl}^* \quad (12)$$

Since q_{ir} is the maximum element of \mathbf{q}_i , we can readily show that $\mathbf{q}_i^T \mathbf{q}_j^* \leq \mathbf{q}_i^T \hat{\mathbf{q}}_j$, which contradicts the assumption shown in equation (7); thus, \mathbf{q}_j^* must be a one-hot vector. Therefore:

$$\mathbf{q}_i^T \mathbf{q}_j^* \leq \max_l \{q_{il}\} \quad (13)$$

Considering (13) and (7), we have:

$$\mathbf{q}_i^T \mathbf{q}_j \leq \max_l \{q_{il}\}. \quad (14)$$

Similarly, for the i th sample, we can show that $\mathbf{q}_i^{*T} \mathbf{q}_j \leq \max_l \{q_{jl}\}$, hence:

$$\mathbf{q}_i^{*T} \mathbf{q}_j \leq \max_l \{q_{jl}\}. \quad (15)$$

(14) and (15) proves (6). □

Corollary 1.1. *If two samples, i and j , are adjacent, the maximum value among their elements is greater than ζ .*

Proof. This statement is a corollary of Theorem 1, that can be proved from (6), (14), (15) and the adjacency definition provided in Definition 3.1. In other words:

$$\zeta \leq \mathbf{q}_i^T \mathbf{q}_j \leq \min(q_{ir}, q_{jo}) \rightarrow \begin{cases} \zeta \leq q_{ir} \\ \zeta \leq q_{jo} \end{cases} \quad (16)$$

where the index of the maximum element of \mathbf{q}_i and \mathbf{q}_j are respectively shown by r and o – i.e., $r = \arg \max_l \{q_{il}\}$ and $o = \arg \max_l \{q_{jl}\}$. \square

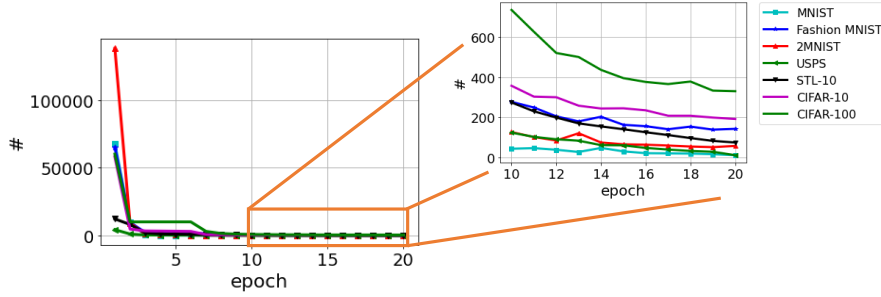


Fig. 3 Number of data points that do not have any adjacent neighbors during the DCSS training in the second phase.

Corollary 1.2. *If a data point has at least one adjacent neighbor, the maximum element of its corresponding \mathbf{q} is greater than ζ .*

Proof. We first empirically test the validity of the employed assumption, i.e. the existence of at least one adjacent (i.e. similar) sample for a data point, on our datasets. Fig. 3 shows the number of data samples that are not similar to any other data points in the \mathbf{q} space. As it can be seen, at the beginning of the second phase of DCSS, since MNet is initialized randomly, many data points do not have any adjacent neighbor (i.e. almost $\forall i, j, i \neq j : \mathbf{q}_i^T \mathbf{q}_j < \zeta$). By minimizing (4) and (5) in the second phase of DCSS, similar samples are tightly packed in the \mathbf{q} space; therefore, almost all samples have at least one adjacent neighbor. For example, only 0.5% of the samples in the CIFAR-100 dataset have no adjacent sample by the end of the training phase. Note that CIFAR-100 presents the worst case among the other datasets shown in Fig 3. All in all, we can roughly assume that each data point has at least one adjacent sample.

Let us consider an arbitrary data point, i , and one of its adjacent data points, j . From Corollary 1.1, we can conclude that:

$$\begin{cases} \zeta \leq \max_l \{q_{il}\} \\ \zeta \leq \max_l \{q_{jl}\} \end{cases} \quad (17)$$

Thus, we proved that the maximum element of \mathbf{q}_i , where i is an arbitrary data point, is greater than ζ . \square

Corollary 1.3. Assume each data point has at least one adjacent neighbor and $\gamma < \zeta^2$. If two data points, i and k , are dissimilar, i and k are not from the same cluster.

Proof. We prove this corollary by contradiction where the contradiction assumption is: i and k are dissimilar, yet from the same cluster where $\gamma < \zeta^2$.

Since i and k are in the same cluster, the index of the maximum element of \mathbf{q}_i and \mathbf{q}_k are the same. (i.e. $\beta = \arg \max_l \{q_{il}\} = \arg \max_l \{q_{kl}\}$). Since each data point has at least one adjacent neighbor, from Corollary 1.2, we can conclude that:

$$\begin{cases} \zeta \leq q_{i\beta} \\ \zeta \leq q_{k\beta} \end{cases} \quad (18)$$

and we can represent $\mathbf{q}_i^T \mathbf{q}_k$ as follow:

$$\mathbf{q}_i^T \mathbf{q}_k = q_{i\beta} q_{k\beta} + \sum_{l \neq \beta} q_{il} q_{kl} \quad (19)$$

Therefore, $q_{i\beta} q_{k\beta} \leq \mathbf{q}_i^T \mathbf{q}_k$. Also, we know i and k are dissimilar. From (18), we can conclude that:

$$\zeta^2 \leq q_{i\beta} q_{k\beta} \leq \mathbf{q}_i^T \mathbf{q}_k \leq \gamma \quad (20)$$

(20) contradicts the assumption of $\gamma < \zeta^2$. Hence, i and k are in different clusters. \square

Theorem 2. For $\frac{2}{3} \leq \zeta$, if i and j are adjacent, they are in the same cluster – i.e. r is equal to o where $r = \arg \max_l \{q_{il}\}$ and $o = \arg \max_l \{q_{jl}\}$.

Proof. First, we find an upper bound for $\mathbf{q}_i^T \mathbf{q}_j$, when the i th and j th samples are adjacent but from different clusters.

Since i and j are not from the same cluster, we can represent $\mathbf{q}_i^T \mathbf{q}_j$ as follows:

$$\begin{aligned} \mathbf{q}_i^T \mathbf{q}_j &= q_{ir} q_{jr} + q_{io} q_{jo} + \sum_{l \neq o, r} q_{il} q_{jl} \\ &\xrightarrow{\forall l: q_{jl} \leq q_{jo}} \leq q_{ir} q_{jr} + q_{io} q_{jo} + \sum_{l \neq o, r} q_{il} q_{jo} \\ &= q_{ir} q_{jr} + q_{jo} \left(\sum_{l \neq r} q_{il} \right) \\ &\xrightarrow{\sum_{l \neq r} q_{il} = 1 - q_{ir}} \leq q_{ir} q_{jr} + q_{jo} (1 - q_{ir}) \\ &\xrightarrow{q_{jr} \leq 1 - q_{jo}} \leq q_{ir} (1 - q_{jo}) + q_{jo} (1 - q_{ir}) \\ &\xrightarrow{\text{from (16)}} \leq q_{ir} (1 - \zeta) + q_{jo} (1 - \zeta) \\ &\xrightarrow{\{q_{ir}, q_{jo}\} \in [0, 1]} \leq 2(1 - \zeta). \end{aligned} \quad (21)$$

Note that $q_{jr} \leq 1 - q_{jo}$ because $\sum_{l \neq o} q_{jl} + q_{jr} + q_{jo} = 1$. Note that all elements of a \mathbf{q} vector are probability values between 0 and 1.

Hence, as is shown (21), if two samples are not from the same cluster, then the inner product of their corresponding \mathbf{q} has an upper bound of $2(1 - \zeta)$. Therefore, if two samples i and j are adjacent (see Definition 1) but from different clusters, then:

$$\begin{aligned} \zeta &\leq \mathbf{q}_i^T \mathbf{q}_j \leq 2(1 - \zeta) \\ \rightarrow \zeta &\leq 2(1 - \zeta) \rightarrow \zeta \leq \frac{2}{3}. \end{aligned} \quad (22)$$

Thus, for $\frac{2}{3} < \zeta$, i and j cannot be from two different clusters. In other words, if two samples i and j are adjacent AND the user-settable parameter ζ is set to a value greater than $\frac{2}{3}$, then the two samples are from similar clusters, i.e., $r = o$. In this paper we set $\zeta = 0.8 > \frac{2}{3}$. \square

Corollary 2.1. Assume $\zeta > \frac{2}{3}$. Consider three data points: i , j , and k . If i and j , and also i and k are adjacent, then j and k are from the same cluster.

Proof. Since i and j (i and k) are adjacent and $\zeta > \frac{2}{3}$, from Theorem 2, we can conclude that i and j (i and k) are in the same cluster; hence, the three samples i , j , and k all are in the same cluster. \square

Theorem 3. Consider three data points i , j , and k where i and j also i and k are adjacent (aka similar). Assume $\zeta > \frac{2}{3}$. If $\gamma < \zeta^2$, then the two samples j and k are not dissimilar (i.e. $\mathbf{q}_j^T \mathbf{q}_k \not\prec \gamma$).

Proof. Considering Theorem 2, i and j and k are in the same cluster. Therefore, we do not want to include the pair of j and k samples as a dissimilar pair when minimizing the loss function defined in equation (5) of the original manuscript.

Since j and k are in the same cluster and knowing that the index of the maximum element in a \mathbf{q} vector shows the cluster of the corresponding sample, we have:

$$\eta = \arg \max_l \{q_{jl}\} = \arg \max_l \{q_{kl}\}. \quad (23)$$

Thus,

$$\mathbf{q}_j^T \mathbf{q}_k = \sum_{l \neq \eta} q_{jl} q_{kl} + q_{j\eta} q_{k\eta}. \quad (24)$$

Therefore,

$$\mathbf{q}_j^T \mathbf{q}_k \geq q_{j\eta} q_{k\eta}. \quad (25)$$

Since $\zeta \leq \mathbf{q}_i^T \mathbf{q}_j$ and $\zeta \leq \mathbf{q}_i^T \mathbf{q}_k$, we can infer (26) from (16).

$$\begin{cases} \zeta \leq q_{j\eta} \\ \zeta \leq q_{k\eta} \end{cases}. \quad (26)$$

From (25) and (26), we can obtain below:

$$\zeta^2 \leq q_{j\eta} q_{k\eta} \leq \mathbf{q}_j^T \mathbf{q}_k. \quad (27)$$

Thus, if we choose $\gamma < \zeta^2$, we will not include the pair of samples j and k as a dissimilar pair in equation (5) of the main manuscript. In this paper, γ is set to 0.2, i.e. $\gamma = 0.2 < 0.8^2$. \square

4 Experiments

In this section, the effectiveness of our proposed DCSS framework is demonstrated on eight benchmark datasets through conducting a rigorous set of experiments. The DCSS clustering performance on the eight benchmark datasets is compared with seventeen clustering methods.

4.1 Datasets

The effectiveness of the proposed method is shown on eight widely used datasets. Considering the unsupervised nature of the clustering task, we concatenate training and test sets when applicable. Combining train and test datasets is a common practice in the clustering research field [37, 13, 15, 14, 16]. The datasets are:

- (1) MNIST [63] consists of 60,000 training and 10,000 test gray-scale handwritten images with size 28×28 . This dataset has ten classes, i.e. $K = 10$.
- (2) Fashion MNIST [64] has the same image size and number of samples as MNIST. However, instead of handwritten images, it consists of different types of fashion products. This makes it fairly more complicated for data clustering compared to the MNIST dataset. It has ten classes of data, i.e. $K = 10$.
- (3) 2MNIST is a more challenging dataset created through the concatenation of the two MNIST and Fashion MNIST datasets. Thus, it has 140,000 gray-scale images from 20 classes, i.e. $K = 20$.
- (4) USPS [65] contains of 9,298 16×16 handwritten images from the USPS postal service. It contains ten classes of data, i.e. $K = 10$.
- (5) CIFAR-10 [66] is comprised of 60,000 RGB images of 10 different items (i.e. $K = 10$), where the size of each image is 32×32 .
- (6) STL-10 [67] is a 10-class image recognition dataset comprising of 13,000 96×96 RGB images. The number of clusters K for this dataset is set to 10.
- (7) CIFAR-100 [66] is similar to the CIFAR-10, except it has 20 super groups based on the similarity between images instead of 10 classes. The number of clusters K for this dataset is set to 20.
- (8) ImageNet-10 comprises ten classes from the larger ImageNet dataset [68]. Despite its reduced size, ImageNet-10 preserves the structural and thematic diversity of the original ImageNet dataset.

The network architecture and implementation details of DCSS for each dataset are presented in Section 1 and Section 2 of the supplementary material file, respectively.

4.2 Evaluation Metrics

We utilize two standard metrics to evaluate clustering performance, including clustering accuracy (ACC) [69] and normalized mutual information (NMI) [70]. ACC finds the best mapping between the true and predicted cluster labels. NMI finds a normalized measure of similarity between two different labels of the same data point. The ACC and NMI formulations are shown below:

$$ACC = \max_{map} \frac{\sum_{i=1}^N \mathbb{1}\{l_i = map(c_i)\}}{N} \quad (28a)$$

$$NMI = \frac{I(l;c)}{\max\{H(l), H(c)\}} \quad (28b)$$

where l_i and c_i denote the true and predicted labels for the data point \mathbf{x}_i . $map(\cdot)$ indicates the best mapping between the predicted and true labels of data points. $I(\mathbf{l}; \mathbf{c})$ denotes the mutual information between true labels $\mathbf{l} = \{l_1, l_2, \dots, l_N\}$ and predicted cluster assignments $\mathbf{c} = \{c_1, c_2, \dots, c_N\}$ for all data points. $H(\cdot)$ presents the entropy function. ACC and NMI range in the interval $[0, 1]$, where higher scores indicate higher clustering performance.

4.3 Clustering Performance

Table 1 ACC and NMI on the benchmark datasets for different clustering methods.

Method \ Datasets	MNIST		Fashion MNIST		2MNIST		USPS		CIFAR-10		STL-10		CIFAR-100		ImageNet-10	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
k-means	53.20	50.00	47.40	51.20	32.31	44.00	65.67	62.00	22.90	8.70	19.20	12.50	13.00	8.40	24.1	11.90
LSSC	71.40	70.60	49.60	49.70	39.77	51.22	63.14	58.94	21.14	10.89	18.75	11.68	14.60	7.92	-	-
LPMF	47.10	45.20	43.40	42.50	34.68	38.69	60.82	54.47	19.10	8.10	18.00	9.60	11.80	7.90	-	-
DEC	84.30	83.72	51.80	54.63	41.20	53.12	75.81	76.91	30.10	25.70	35.90	27.60	18.50	13.60	38.10	28.20
IDEC	88.13	83.81	52.90	55.70	40.42	53.56	75.86	77.68	36.99	32.53	32.53	18.85	19.61	14.58	39.40	29.00
DCN	83.00	81.00	51.22	55.47	41.35	46.89	73.00	71.90	30.47	24.58	33.84	24.12	20.17	12.54	37.40	27.30
DKM	84.00	81.54	51.31	55.57	41.75	46.58	75.70	77.60	35.26	26.12	32.61	29.12	18.14	12.30	38.10	29.80
VaDE	94.50	87.60	50.39	59.63	40.35	58.37	56.60	51.20	29.10	24.50	28.10	20.00	15.20	10.80	33.40	19.30
DAC	97.75*	93.51*	62.80	58.90	-	-	-	-	52.18	39.59	46.99	36.56	23.75	18.52	52.70	39.40
GANMM	64.00	61.00	34.00	27.00	-	-	50.12	49.35	-	-	-	-	-	-	-	-
CC	88.56	84.21	64.52	61.45	42.15	58.89	81.21	79.45	77.00	67.80	85.00	76.40	42.30	42.10	89.30	85.00
PICA	-	-	-	-	-	-	-	-	69.60	59.10	71.30	61.10	33.70	31.00	87.00	80.20
EDESC	91.30	86.20	63.10*	67.00*	-	-	-	-	62.70	46.40	74.50	68.70	38.50	37.00	-	-
IDFD	-	-	-	-	-	-	-	-	81.50	71.10	75.60	64.30	42.50	42.60	95.40*	89.80*
MICE	-	-	-	-	-	-	-	-	83.50*	73.70*	75.20	63.50	44.00	43.60	-	-
DCCM	-	-	-	-	-	-	68.60	67.50	62.30	49.60	48.20	37.60	32.70	28.50	71.00	60.80
AE + k-means	86.03	80.25	57.94	57.15	44.01	62.80	75.11	74.45	80.23	68.65	85.29	76.00	43.81	42.86	87.23	85.22
DCSS _u	95.99	89.95	62.90	63.58	45.31*	63.00*	82.93*	81.84*	82.43	71.32	86.47*	76.52*	44.08*	43.70*	89.46	86.30
DCSS	98.00	94.71	66.40	67.10	48.57	67.80	87.21	86.10	84.16	75.09	87.91	77.39	45.10	44.51	95.70	90.52

The effectiveness of our proposed DCSS method is compared against seventeen well-known algorithms, including conventional and state-of-the-art deep-learning-based clustering methods, using the commonly used evaluation metrics ACC and NMI, defined in Section 4.2.

The conventional clustering methods are k-means [7], large-scale spectral clustering (LSSC) [71], and locality preserving non-negative matrix factorization (LPMF) [72]. Deep learning-based algorithms are deep embedding clustering (DEC) [37], improved deep embedding clustering (IDEC) [15], deep clustering network (DCN) [14], deep k-means (DKM) [13], variational deep embedding (VaDE) [73], GAN mixture model for

clustering (GANMM) [74], deep adaptive clustering (DAC) [27], and the very recent clustering methods such as contrastive clustering (CC) [23], deep semantic clustering by partition confidence maximization (PICA) [75], efficient deep embedded subspace clustering (EDESC) [76], instance discrimination and feature decorrelation (IDFD) [77], Mixture of contrastive experts for unsupervised image clustering (MICE) [78], and deep comprehensive correlation mining (DCCM) [28]. In addition, we report the clustering performance of a baseline method AE + k-means in which k-means is simply applied to the latent representation of an AE that has a similar architecture as the AE used in the DCSS method, trained based on minimizing the dataset reconstruction loss. More details about the comparing algorithms can be found in Section 2.

We also demonstrate the success of the first phase of DCSS, presented in Section 3.1, in creating the reliable subspace \mathbf{u} in which the data points form hypersphere-like clusters around their corresponding cluster center. To this end, we only implement the first phase of the DCSS algorithm – i.e., we train the DCSS’s AE through minimizing the loss function presented in (1), where the AE architecture and its initialization are similar to those presented in Section 1 of the supplementary material file. After training the \mathbf{u} space, we perform a crisp cluster assignment by considering each data hypersphere-like group in the \mathbf{u} space as a data cluster and assigning each data point to the one with the closest center. In the following tables and figures, clustering using only the first phase is shown as DCSS_u. A preliminary version of DCSS_u is presented in [79].

The clustering performance of DCSS_u and DCSS, along with the comparison algorithms, are shown in Table 1. For the comparison methods, if the ACC and NMI of a dataset are not reported in the corresponding original paper, we ran the released code with the same hyper-parameters discussed in the original paper. When the code is not publicly available or not applicable to the dataset, we put dash marks (-) instead of the corresponding results. The best result for each dataset is shown in bold. The second top results are shown with *.

Several observations can be made from Table 1: (1) The proposed DCSS method outperforms all of our comparison methods on all datasets. (2) The first phase of DCSS (shown as DCSS_u) effectively groups the data points around their corresponding centers. This can be inferred from DCSS_u’s ACC and NMI values. When measuring ACC, DCSS_u outperforms other methods in five out of seven datasets, and it exhibits even stronger performance by outperforming competitors in six out of seven datasets when considering NMI. (3) Effectiveness of the self-supervision with similar and dissimilar pairs of samples can be inferred by comparing DCSS with DCSS_u. It can be seen that DCSS significantly outperforms DCSS_u on all datasets. (4) Effectiveness of the AE’s loss function proposed in equation (1) compared to the case of training AE with only the reconstruction loss can be inferred by comparing the DCSS_u performance with the baseline method AE+k-means. As can be seen, the DCSS_u clearly outperforms AE+k-means on all datasets.

4.4 Effectiveness of the First Phase

To demonstrate the efficacy of introducing cluster-specific losses and the adopted approach in iteratively updating the network’s parameters over K successive runs, as

Table 2 ACC and NMI on the benchmark datasets for different methods.

Datasets Method	MNIST		Fashion MNIST		2MNIST		USPS		CIFAR-10		STL-10		CIFAR-100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
AE + k-means	86.03	80.25	57.94	57.15	44.01	62.80	75.11	74.45	80.23	68.65	85.29	76.00	43.81	42.86
DCSS _{agg}	87.51	81.35	59.61	58.47	45.21	62.90	76.38	75.17	81.39	70.77	85.60	76.10	43.71	42.90
DCSS _u	95.99	89.95	62.90	63.58	45.31	63.00	82.93	81.84	82.43	71.32	86.47	76.52	44.08	43.70

is discussed in Section 3.1, we present a comparative analysis against an alternative approach where all loss terms are aggregated in the initial phase, followed by a single backward pass to update the network parameters at once. The results are presented in Table 2 as DCSS_{agg}, highlighting a consistent trend. Across all experiments and datasets, the performance of DCSS_u surpasses that of DCSS_{agg} and AE + k-means, showing the advantages of employing cluster-specific loss functions and iteratively updating the network parameters in K successive runs.

4.5 t-SNE visualization

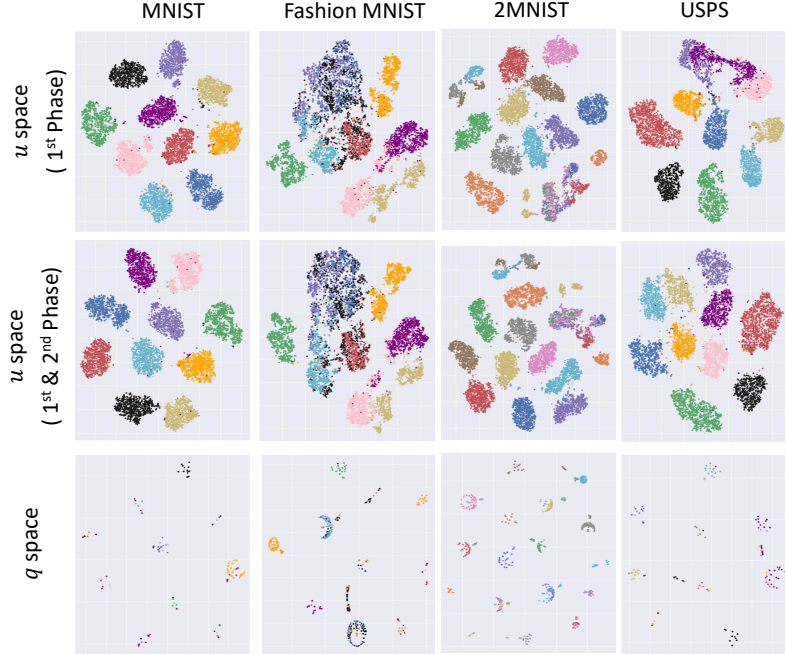


Fig. 4 Clustering visualization of different phases of DCSS using t-SNE for different benchmark datasets. For reference, the visualization for the baseline model AE+k-means is shown in the first row. Axes range from -100 to 100.

Fig. 4 illustrates the effectiveness of different phases of our proposed DCSS framework for all datasets, where t-SNE [80] is used to map the output of DCSS’s encoder and MNet to a 2D space. The different colors correspond to the different data clusters.

The first row of Fig. 4 shows the representation of different data points in the \mathbf{u} space, i.e. the latent space of the DCSS’s AE, only after completing the first phase discussed in Section 3.1. As it can be seen, after completing the first phase of DCSS, different clusters of data points are fairly separated, sit near their corresponding centers, and form spheres; however, not all clusters are well separated. For example, in the USPS dataset, the data clusters shown in pink, purple, and magenta are mixed together. This indicates the insufficiency of the reconstruction and centering losses for the clustering task.

The second row of Fig. 4 shows the data representations in the \mathbf{u} space after completing the second phase of DCSS discussed in Section 3.2, where \mathbf{u} is refined by minimizing (4) and (5). As it can be seen, refining the \mathbf{u} space employing pairwise similarities results in more **dense** and separate cluster distributions. For example, the pink, purple, and magenta clusters of USPS are now well distinguishable in the new refined \mathbf{u} space. As another example, see samples of the three clusters shown in red, olive, and brown of the 2MNIST dataset. These clusters are more separable in the refined \mathbf{u} space compared to the corresponding representation shown in the first row.

The last row in Fig. 4 depicts the output space of MNet (i.e. the \mathbf{q} space), in which we make decisions about final cluster assignments of data points. As is expected, clusters in this space have low within- and high between-cluster distances, and cluster distributions can take non-hypersphere patterns. As an example, consider the cyan and the purple clusters of the Fashion MNIST. These clusters are mixed in the \mathbf{u} space, but they are completely isolated in the \mathbf{q} space.

4.6 Effect of input features

To make sure our improved performance isn’t solely due to using features from SimCLR as input for our algorithm, we conducted an additional experiment which presents the performance of the other deep learning-based methods that can accept the extracted features, here SimCLR, as the input of their algorithm. To this end, we evaluated the clustering performance using two types of features from SimCLR. In the first experiment, named SimCLR (Z space), we used the output of the projection head as input for DCSS and the other algorithms. In the second experiment, SimCLR (H space), we used the output of ResNet-34 (i.e. the SimCLR’s backbone), after the average pooling layer, as input. The results are shown in Table 3. The best results for each dataset are in bold, and the second-best results are marked with an asterisk (*). Notably, our DCSS method consistently outperforms all other algorithms across all datasets. Additionally, DCSS_u consistently ranks as the second-best method in all experiments.

4.7 Loss function convergence

Fig. 5. depicts the average, over different clusters on different batches of data points, of the reconstruction, centering, and total losses corresponding to the first phase of DCSS (i.e. DCSS_u) shown in (1). As can be seen, all losses are converged at the end of training. The noticeable reduction in the centering loss shows the effectiveness of our proposed approach in creating a reliable \mathbf{u} space in which the data points are gathered

Table 3 ACC and NMI with different extracted features using SimCLR

Dataset	Method	SimCLR (Z space)		SimCLR (H space)	
		ACC	NMI	ACC	NMI
STL-10	DEC	84.26	75.14	85.66	76.13
	IDEC	84.55	75.64	85.90	76.44
	DCN	81.26	71.23	82.84	73.35
	DKM	83.14	74.93	83.64	74.85
	AE+k-means	84.59	75.39	85.29	76.00
	DCSS _u	85.11*	76.31*	86.47*	76.52*
	DCSS	86.66	76.66	87.91	77.39
CIFAR-10	DEC	76.60	66.91	81.59	70.00
	IDEC	76.78	67.02	81.89	70.49
	DCN	73.00	62.35	78.05	67.82
	DKM	76.99	66.52	81.19	69.47
	AE+k-means	75.13	65.98	80.23	68.65
	DCSS _u	77.21*	68.00*	82.43*	71.32*
	DCSS	77.55	68.91	84.16	75.09
CIFAR-100	DEC	42.42	42.22	43.75	42.55
	IDEC	41.83	41.20	43.87	42.73
	DCN	40.88	40.13	42.11	41.59
	DKM	42.36	42.16	43.40	42.90
	AE+k-means	42.56	42.26	43.81	42.86
	DCSS _u	43.12*	42.88*	44.08*	43.70*
	DCSS	43.43	43.18	45.10	44.51

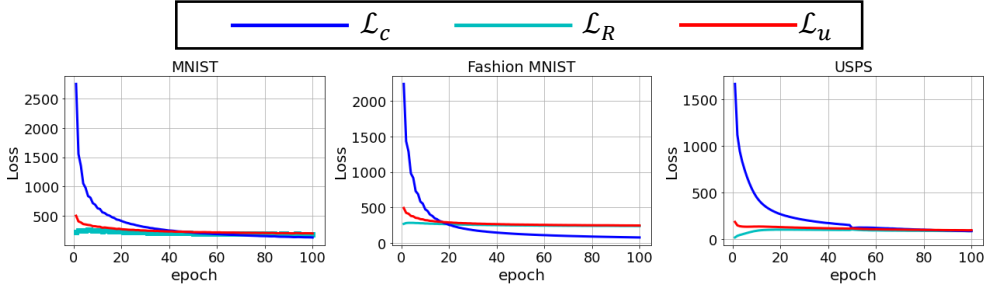


Fig. 5 The reconstruction loss \mathcal{L}_r , centering loss \mathcal{L}_c , and total loss \mathcal{L}_u of the first phase of DCSS vs. training epochs, for different datasets.

around the centers. Moreover, the figures show that at the first training epochs, our method trades the reconstruction loss for improved centering performance. This proves the insufficiency of the reconstruction loss in creating a reliable latent space for data clustering.

In Fig. 6, we investigate the convergence of the second phase losses, shown in equations (4) and (5). Since we initialize the MNet randomly, at the first few epochs,

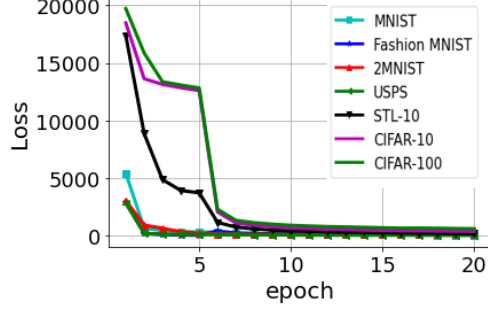


Fig. 6 The second phase’s loss function of DCSS for different benchmark datasets.

MNet has little knowledge about the lower-dimension representation of the data points in the \mathbf{q} space; thus, we face a high loss value. As the training process progresses, the loss value drops and converges to zero at the end of the training process. In the first T_2 epochs ($T_2 = 5$), the algorithm minimizes the loss presented in (4). It minimizes (5) in the remaining epochs. The continuity of the loss reduction over epochs, along with the sharp loss drop at the 5th epoch, confirms the effectiveness of our proposed strategy in employing \mathbf{u} for similarity measurements in the early epochs and then \mathbf{q} in the later epochs.

4.8 DCSS as a General Framework

Table 4 ACC and NMI on the benchmark datasets when employing DCSS as a general framework to improve state-of-the-art AE-based clustering methods.

Method \ Datasets	MNIST		Fashion MNIST		2MNIST		USPS		CIFAR-10		STL-10		CIFAR-100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
DEC+MNet	89.13	86.97	61.25	56.30	44.25	57.35	77.58	78.15	81.88	70.79	85.97	76.33	44.10	42.81
IDEK+MNet	90.51	85.42	60.12	57.16	44.83	58.00	76.58	78.14	82.26	70.86	86.11	76.35	44.21	43.01
DCN+MNet	87.49	83.25	54.23	58.69	45.62	48.24	76.90	77.59	79.13	68.51	83.11	74.00	43.21	41.86
DKM+MNet	88.31	84.52	57.23	56.26	44.34	49.50	77.13	78.02	81.30	70.63	83.75	75.01	43.78	43.19

In this section, we demonstrate the effectiveness of the DCSS method as a general framework where the \mathbf{u} space is trained with other AE-based clustering techniques. To this end, we substitute the first phase, presented in Section 3.1, with other deep learning-based techniques that train an effective subspace using an AE for the purpose of data clustering. Among our comparison methods, DEC, IDEK, DCN, and DKM algorithms are AE-based. For each dataset, we train AEs using these algorithms, then take their encoder part and append our proposed MNet to the latent space. Then, we run the second phase of DCSS. Results of such implementation are reported in Table 4 where X+MNet indicates the performance of DCSS employing the X method’s latent space as the DCSS’s \mathbf{u} space. Note that, for the RGB datasets, features are constructed using the pre-trained SimCLR (H space).

Comparing the clustering results reported in Table 1, Table 3, and Table 4 confirms the effectiveness of DCSS as a general framework to improve the existing state-of-the-art AE-based clustering methods. On average, MNet improves the clustering

performance of DEC, IDEC, DCN, and DKM respectively by 2.68% (1.66%), 2.23% (1.21%), 2.58% (1.35%), and 2.12% (1.23%) in terms of ACC (NMI).

4.9 Performance on Imbalanced Dataset

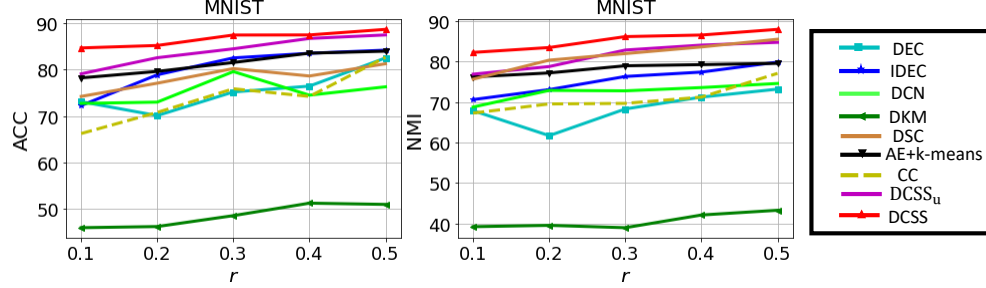


Fig. 7 The clustering performance of different methods on imbalanced samples of MNIST.

To demonstrate the effectiveness of our proposed DCSS method on an imbalanced dataset, we randomly collect five subsets of the MNIST dataset with different retention rates $r \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, where samples of the first class are chosen with the probability of r and the last class with probability of 1, with the other classes linearly in between. Hence, on average, the number of samples for the first cluster is r times less than that of the last cluster. As is shown in Fig. 7, our proposed DCSS framework significantly outperforms our comparison methods for all r values. This indicates the robustness of DCSS on imbalanced data. As is expected, in general, for all methods, increasing r results in a higher performance because the dataset gets closer to a balanced one. Higher performance of DCSS on imbalanced datasets can be associated with two factors: (1) considering an individual loss for every cluster in the 1st phase, and (2) considering the pairwise data relations.

4.10 Visualization of \mathbf{q} vectors

Fig. 8 shows the representations of the data points from various clusters in the \mathbf{q} space. As can be seen, the proposed DCSS method results in representations that are very close to the one-hot vectors. Note that the k th element of \mathbf{q}_i denotes the probability of sample \mathbf{x}_i being in the k th data cluster. The closer \mathbf{q}_i is to the one-hot vector, the more confidently a crisp cluster assignment can be made. As is proved in Corollary 1.2 of Section 3.4, if data point \mathbf{x}_i has at least one similar neighbor, the maximum element of \mathbf{q}_i is greater than ζ . In our experiments, ζ is set to 0.8. This can justify the aggregation of the data points near the one-hot vectors in the \mathbf{q} space.

To further demonstrate convergence of the \mathbf{q} representations to one-hot vectors, histogram of residuals $h_i = \|\mathbf{I}_i - \mathbf{q}_i\|_1$, $i = 1, \dots, N$ for all datasets are shown in Fig. 9; where $\|\cdot\|_1$ indicates the ℓ_1 -norm and \mathbf{I}_i is the one-hot crisp assignment corresponding to \mathbf{q}_i – i.e. the index of the non-zero element of \mathbf{I}_i is equal to the index of the maximum

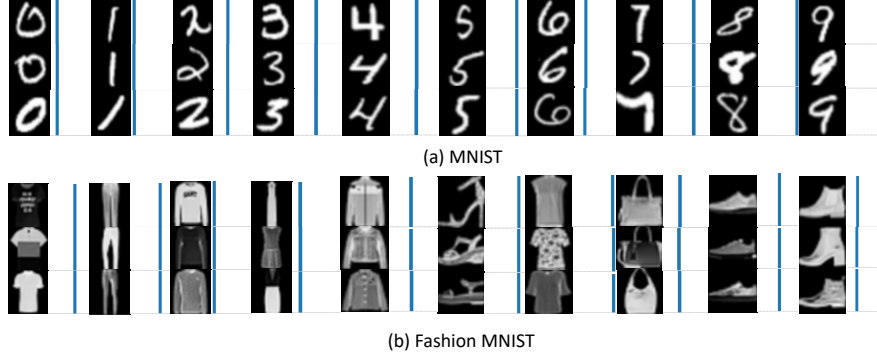


Fig. 8 Visualization of \mathbf{q} for samples from (a) MNIST and (b) Fashion MNIST datasets. The \mathbf{q} vector for each image is depicted beside the image. The vertical axes range from 0 to 1.

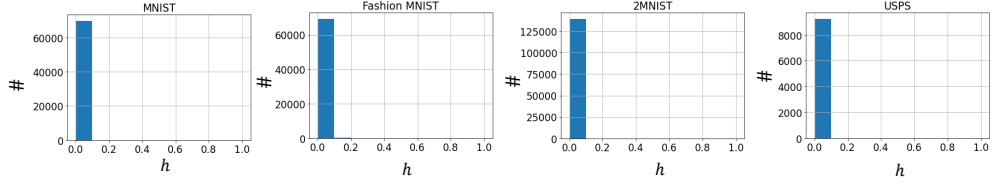


Fig. 9 Histogram plot of $h_i = \|\mathbf{I}_i - \mathbf{q}_i\|_1$, $i = 1, \dots, N$ where \mathbf{I}_i is the one-hot crisp assignment corresponding to \mathbf{q}_i .

element of \mathbf{q}_i . As can be seen in Fig. 9, the representation of almost all data points in the \mathbf{q} space is very close to their corresponding one-hot vector.

4.11 Hyperparameters Sensitivity

In Fig. 10, we investigate the effect of different hyperparameters on DCSS clustering performance. For hyperparameters of the first phase (i.e. α , m , and T_1), we report the performance of clustering using DCSS_u (as is presented in Section 4.3).

In our proposed method, hyperparameters are fixed across all datasets, i.e. no fine-tuning is performed per dataset. Hence, one may obtain more accurate results by tuning the hyperparameters per dataset.

In Fig. 10(a), we explore the importance of the centering loss in the first phase’s loss function, shown in (1), by changing $\alpha \in \{0, 0.01, 0.1, 1\}$ for MNIST dataset. As is shown in this figure, by increasing the value of α from 0 to 0.1, our DCSS performance significantly enhanced in terms of ACC and NMI, which demonstrates the effectiveness of incorporating the centering loss beside the reconstruction loss in the first phase’s loss function. We observed a similar trend across all the other datasets. In all our experiments, for all datasets, α is set to 0.1.

Fig. 10(b) shows the impact of the level of fuzziness m on the clustering performance of DCSS_u for the Fashion MNIST dataset, where $m \in \{1.1, 1.3, 1.5, 1.7\}$. In the

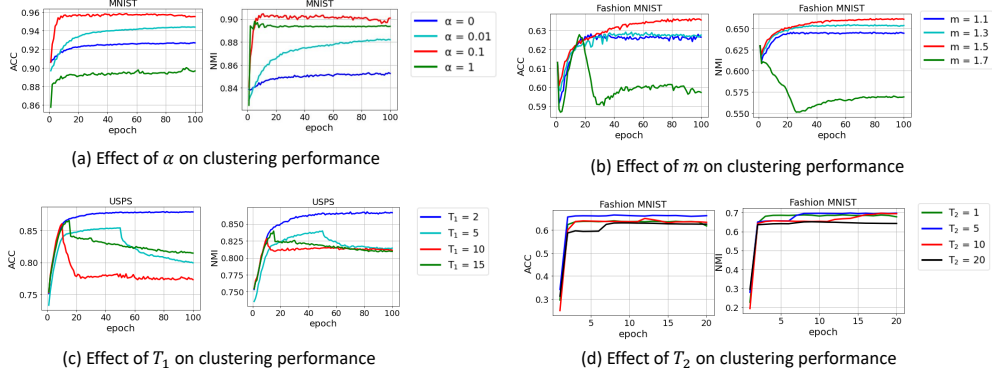


Fig. 10 Sensitivity of DCSS to different hyperparameters.

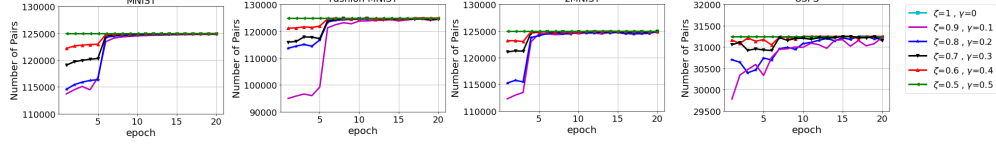
case of $m \rightarrow 1$ ($m \rightarrow \infty$), group membership vectors converge to one-hot (equal probability) vectors. As shown in this figure, as desired, the DCSS method is not too sensitive to m when it is set to a reasonable value. We observed a consistent pattern across all the remaining datasets. In all our experiments, for all datasets, m is set to 1.5.

In Fig. 10(c), we scrutinize the effect of update interval T_1 in the clustering performance of the first phase for $T_1 \in \{2, 5, 10, 15\}$. As is expected, better clustering performance in terms of ACC and NMI is acquired for a smaller value of T_1 for the USPS dataset. Consistently, the same behavior was observed across all remaining datasets. In our experiments, for all datasets, T_1 is set to 2.

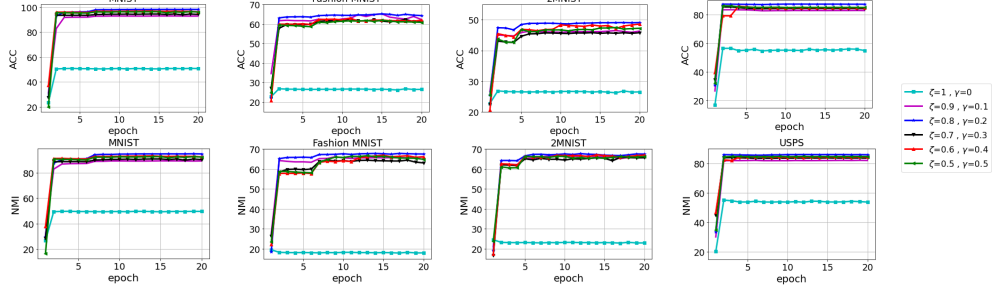
In Fig. 10(d), we change the number of training epochs T_2 , defined in Section 3.2, for Fashion MNIST dataset, where $T_2 \in \{1, 5, 10, 20\}$. As is expected, for a very small T_2 value, e.g. $T_2 = 1$, where training the \mathbf{q} space is mainly supervised by the \mathbf{q} space itself even at the MNet training outset, DCSS cannot provide a proper \mathbf{q} space, since \mathbf{q} is not a sufficiently reliable space to be used for self-supervision. The figure also shows that for a very large T_2 value, e.g. $T_2 = 20$, when we only trust the \mathbf{u} space for supervising the \mathbf{q} space, we cannot train an effective \mathbf{q} space. As shown, a good clustering performance can be obtained when T_2 is set to a moderate value. In our experiments, for all datasets, T_2 is set to 5. This demonstrates the effectiveness of the proposed strategy in supervising the MNet training using both the \mathbf{u} and \mathbf{q} spaces.

In Fig. 11, we change ζ and γ in range $[0, 1]$, where $\zeta + \gamma = 1$, to observe model convergence and accuracy for different lengths of the ambiguity interval, defined as $\zeta - \gamma$, ranging from 1 (when $\zeta = 1$) to 0 (when $\zeta = 0.5$). Fig. 11(a) shows the number of pairs participating in minimizing the loss functions defined in (4) and (5). As can be seen, at the beginning of the second phase, our model can make a decisive decision only about a few pairs, and the remaining pairs are in the ambiguous region. As the second phase of the training process progresses, more and more pairs are included in the loss functions optimization process. Finally, at the end of the second phase, almost all pairs contribute to the training.

Furthermore, in Fig. 11(b), we investigate the influence of ζ and γ in clustering performance. As it can be seen, as is desired, the final clustering performance of our



(a) Number of data samples participating in the second phase of DCSS



(b) ACC and NMI for different γ and ζ

Fig. 11 Changing hyperparameters ζ and γ for different datasets. (a) Number of data pairs participating in the second phase of DCSS. (b) Clustering performance in terms of ACC and NMI for different datasets for different values of ζ and γ ; the clustering performance of DCSS is less sensitive to the choice of ζ and γ in the range of $[0.5, 0.9]$ and $[0.1, 0.5]$, respectively.

DCSS framework is not highly sensitive to the choice of ζ and γ when these are set to reasonable values. In all our experiments, $\zeta = 0.8$ and $\gamma = 0.2$ for all experiments and datasets.

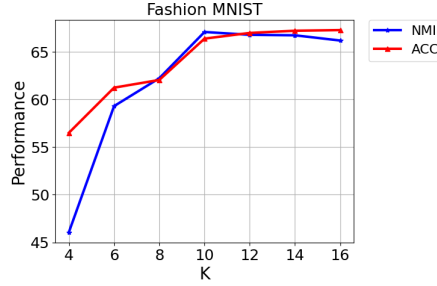


Fig. 12 Sensitivity of DCSS model to the initial number of clusters (K).

In Figure 12, we investigate the impact of initializing the number of clusters to values smaller or larger than the actual number of clusters on the performance of our DCSS model using the Fashion MNIST dataset, where the true number of clusters is 10. When the initial number of clusters (e.g., $K = 4$) is smaller than the actual number, the model is forced to group data samples from multiple clusters into a single

cluster, resulting in a decrease in the DCSS model’s performance in terms of ACC and NMI. Conversely, with higher values of K (e.g., $K = 16$), the algorithm may tend to over-segment the data, creating smaller clusters that may not well align with the true structure of the data. In an extreme scenario where K is set to a very large value, the over-segmenting of the data may cause overfitting to the training data. Employing a validation set becomes valuable when conducting a parameter sweep to identify an appropriate value for K . See [81, 82] for more discussion on how to have a prior estimation for the number of clusters.

4.12 Features visualization

In order to investigate the effectiveness of our model in extracting useful features for different datasets, we train a deep neural network with the same structure as is presented in Section 1 of the supplementary material file in a *supervised* manner, and then we compare the output of the first convolutional layers for the trained model and our proposed DCSS model. As it can be seen in Fig. 13, our DCSS learns a variety of low- and high-frequency features, which are similar to features learned in a supervised manner. This demonstrates the effectiveness of our framework in finding informative features in an unsupervised manner.

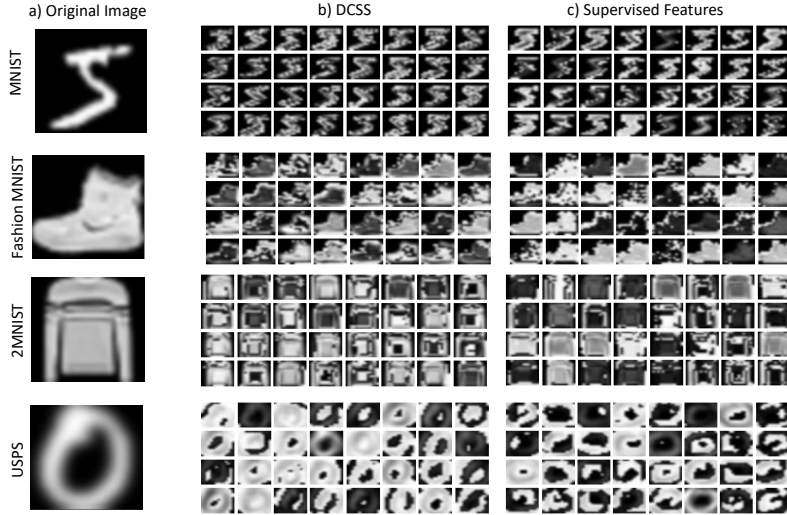


Fig. 13 (a) samples of MNIST, Fashion MNIST, 2MNIST, and USPS. The output of the first convolutional layer using (b) the unsupervised DCSS method and (c) a supervised manner employing the same network structure as of DCSS is shown in Section 1 of the supplementary material file.

5 Conclusion

In this paper, we present a novel, effective, and practical method for data clustering employing the novel concept of self-supervision with pairwise data similarities. Despite

most clustering methods, the proposed DCSS algorithm employs soft cluster assignments in its loss function, optimizes cluster-specific losses, and takes advantage of the relevant information available in the sample pairs. The proposed algorithm is shown to perform well in practice compared to previous state-of-the-art clustering algorithms. We also show that the DCSS’s self-supervision approach can be employed as a general approach to improve the performance of state-of-the-art AE-based clustering methods.

References

- [1] John R Hershey et al. “Deep clustering: Discriminative embeddings for segmentation and separation”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 31–35.
- [2] Xia Hu, Qiaoyu Tan, and Ninghao Liu. “Deep representation learning for social network analysis”. In: *Frontiers in Big Data* 2 (2019), p. 2.
- [3] Mei Wang and Weihong Deng. “Deep face recognition with clustering based domain adaptation”. In: *Neurocomputing* (2020).
- [4] Wenbin Zhu et al. “Local-adaptive face recognition via graph-based meta-clustering and regularized adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20301–20310.
- [5] Mathilde Caron et al. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149.
- [6] Yu Kong and Yun Fu. “Human action recognition and prediction: A survey”. In: *International Journal of Computer Vision* 130.5 (2022), pp. 1366–1401.
- [7] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.
- [8] Seunghyoung Ryu et al. “Convolutional autoencoder based feature extraction and clustering for customer load analysis”. In: *IEEE Transactions on Power Systems* 35.2 (2019), pp. 1048–1060.
- [9] Chunfeng Song et al. “Auto-encoder based data clustering”. In: *Iberoamerican congress on pattern recognition*. Springer. 2013, pp. 117–124.
- [10] Pierre Baldi. “Autoencoders, unsupervised learning, and deep architectures”. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings. 2012, pp. 37–49.
- [11] Junaid Haseeb et al. “Autoencoder-based feature construction for IoT attacks clustering”. In: *Future Generation Computer Systems* 127 (2022), pp. 487–502.
- [12] Mohammadreza Sadeghi and Narges Armanfard. “IDECF: Improved deep embedding clustering with deep fuzzy supervision”. In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 1009–1013.
- [13] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. “Deep k-means: Jointly clustering with k-means and learning representations”. In: *Pattern Recognition Letters* 138 (2020), pp. 185–192.

- [14] Bo Yang et al. “Towards k-means-friendly spaces: Simultaneous deep learning and clustering”. In: *international conference on machine learning*. 2017, pp. 3861–3870.
- [15] Xifeng Guo et al. “Improved deep embedded clustering with local structure preservation.” In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 1753–1759.
- [16] Xu Yang et al. “Deep spectral clustering using dual autoencoder network”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4066–4075.
- [17] Mohammad Reza Sadeghi and Narges Armanfard. “IDECF: Improved Deep Embedding Clustering With Deep Fuzzy Supervision”. In: *ICIP 2021*. 2021.
- [18] Yaniv Opochninsky et al. “K-autoencoders deep clustering”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 4037–4041.
- [19] Mathilde Caron et al. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149.
- [20] Kai Tian, Shuigeng Zhou, and Jihong Guan. “Deepcluster: A general clustering framework based on deep learning”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pp. 809–825.
- [21] Jing Pan et al. “Image deep clustering based on local-topology embedding”. In: *Pattern Recognition Letters* 151 (2021), pp. 88–94.
- [22] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.
- [23] Yunfan Li et al. “Contrastive clustering”. In: *2021 AAAI Conference on Artificial Intelligence (AAAI)*. 2021.
- [24] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.
- [25] Mathilde Caron et al. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *Advances in neural information processing systems* 33 (2020), pp. 9912–9924.
- [26] Mohammadreza Sadeghi, Hadi Hojjati, and Narges Armanfard. “C3: Cross-instance guided contrastive clustering”. In: *arXiv preprint arXiv:2211.07136* (2022).
- [27] Jianlong Chang et al. “Deep adaptive image clustering”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5879–5887.
- [28] Jianlong Wu et al. “Deep comprehensive correlation mining for image clustering”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 8150–8159.
- [29] Jianlong Chang et al. “Deep discriminative clustering analysis”. In: *arXiv preprint arXiv:1905.01681* (2019).
- [30] Wouter Van Gansbeke et al. “Scan: Learning to classify images without labels”. In: *Proceedings of the European Conference on Computer Vision*. 2020.

- [31] Mahmut Kaya and Hasan Şakir Bilge. “Deep metric learning: A survey”. In: *Symmetry* 11.9 (2019), p. 1066.
- [32] Jian Wang et al. “Deep metric learning with angular loss”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2593–2601.
- [33] Eric P Xing et al. “Distance metric learning with application to clustering with side-information”. In: *NIPS*. Vol. 15. 505–512. Citeseer. 2002, p. 12.
- [34] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.
- [35] Yutong Xie et al. “Pairwise relation learning for semi-supervised gland segmentation”. In: *Medical Image Computing and Computer Assisted Intervention—MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part V 23*. Springer. 2020, pp. 417–427.
- [36] Jerome H Friedman. “On bias, variance, 0/1—loss, and the curse-of-dimensionality”. In: *Data mining and knowledge discovery* 1.1 (1997), pp. 55–77.
- [37] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised deep embedding for clustering analysis”. In: *International conference on machine learning*. 2016, pp. 478–487.
- [38] Dongkuan Xu and Yingjie Tian. “A comprehensive survey of clustering algorithms”. In: *Annals of Data Science* 2.2 (2015), pp. 165–193.
- [39] Mohammadreza Sadeghi and Narges Armanfard. “Deep Multirepresentation Learning for Data Clustering”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).
- [40] Chuang Niu, Hongming Shan, and Ge Wang. “SPICE: Semantic Pseudo-Labeling for Image Clustering”. In: *IEEE Transactions on Image Processing* 31 (2021), pp. 7264–7278. URL: <https://api.semanticscholar.org/CorpusID:232258051>.
- [41] Yuming Shen et al. “You never cluster alone”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27734–27746.
- [42] Zhizhong Huang et al. “Learning representation for clustering via prototype scattering and positive sampling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
- [43] Weihua Hu et al. “Learning discrete representations via information maximizing self-augmented training”. In: *International conference on machine learning*. PMLR. 2017, pp. 1558–1567.
- [44] Sungwon Park et al. “Improving Unsupervised Image Clustering With Robust Learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12278–12287.
- [45] Jianlong Chang et al. “Deep Self-Evolution Clustering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2020), pp. 809–823.
- [46] Yunfan Li et al. “Twin contrastive learning for online clustering”. In: *International Journal of Computer Vision* 130.9 (2022), pp. 2205–2221.
- [47] Erxue Min et al. “A survey of clustering with deep learning: From the perspective of network architecture”. In: *IEEE Access* 6 (2018), pp. 39501–39514.

- [48] Peihao Huang et al. “Deep embedding network for clustering”. In: *2014 22nd International conference on pattern recognition*. IEEE. 2014, pp. 1532–1537.
- [49] Fei Tian et al. “Learning deep representations for graph clustering”. In: *28th AAAI Conference on Artificial Intelligence*. 2014.
- [50] Satu Elisa Schaeffer. “Graph clustering”. In: *Computer science review* 1.1 (2007), pp. 27–64.
- [51] Maria CV Nascimento and Andre CPLF De Carvalho. “Spectral methods for graph clustering—a survey”. In: *European Journal of Operational Research* 211.2 (2011), pp. 221–231.
- [52] Andrew Y Ng, Michael I Jordan, and Yair Weiss. “On spectral clustering: Analysis and an algorithm”. In: *Advances in neural information processing systems*. 2002, pp. 849–856.
- [53] Santo Fortunato. “Community detection in graphs”. In: *Physics reports* 486.3-5 (2010), pp. 75–174.
- [54] Uri Shaham et al. “Spectralnet: Spectral clustering using deep neural networks”. In: *arXiv preprint arXiv:1801.01587* (2018).
- [55] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).
- [56] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).
- [57] Zhuxi Jiang et al. “Variational deep embedding: An unsupervised and generative approach to clustering”. In: *arXiv preprint arXiv:1611.05148* (2016).
- [58] Warith Harchaoui, Pierre-Alexandre Mattei, and Charles Bouveyron. “Deep adversarial Gaussian mixture auto-encoder for clustering”. In: (2017), pp. 1–5.
- [59] Alireza Makhzani et al. “Adversarial Autoencoders”. In: *International Conference on Learning Representations*. 2016. URL: <http://arxiv.org/abs/1511.05644>.
- [60] Wenming Cao et al. “Unsupervised discriminative feature learning via finding a clustering-friendly embedding space”. In: *Pattern Recognition* 129 (2022), p. 108768.
- [61] Xi Chen et al. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2016, pp. 2180–2188.
- [62] Yang Yu and Wen-Ji Zhou. “Mixture of GANs for Clustering.” In: *IJCAI*. 2018, pp. 3047–3053.
- [63] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [64] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).
- [65] Jonathan J. Hull. “A database for handwritten text recognition research”. In: *IEEE Transactions on pattern analysis and machine intelligence* 16.5 (1994), pp. 550–554.

- [66] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).
- [67] Adam Coates, Andrew Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 215–223.
- [68] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.
- [69] Yi Yang et al. “Image clustering using local discriminant models and global integration”. In: *IEEE Transactions on Image Processing* 19.10 (2010), pp. 2761–2773.
- [70] Wei Xu, Xin Liu, and Yihong Gong. “Document clustering based on non-negative matrix factorization”. In: *26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. 2003, pp. 267–273.
- [71] Xinlei Chen and Deng Cai. “Large scale spectral clustering with landmark-based representation”. In: *25th AAAI conference on artificial intelligence*. 2011.
- [72] Deng Cai et al. “Locality preserving nonnegative matrix factorization”. In: *Twenty-first international joint conference on artificial intelligence*. 2009.
- [73] Zhuxi Jiang et al. “Variational deep embedding: An unsupervised and generative approach to clustering”. In: *arXiv preprint arXiv:1611.05148* (2016).
- [74] Yang Yu and Wen-Ji Zhou. “Mixture of GANs for Clustering.” In: *IJCAI*. 2018, pp. 3047–3053.
- [75] Jiabo Huang, Shaogang Gong, and Xiatian Zhu. “Deep semantic clustering by partition confidence maximisation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8849–8858.
- [76] Jinyu Cai et al. “Efficient deep embedded subspace clustering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1–10.
- [77] Yaling Tao, Kentaro Takagi, and Kouta Nakata. “Clustering-friendly representation learning via instance discrimination and feature decorrelation”. In: *International Conference on Learning Representations (ICLR)* (2021).
- [78] Tsung Wei Tsai, Chongxuan Li, and Jun Zhu. “Mice: Mixture of contrastive experts for unsupervised image clustering”. In: *International conference on learning representations*. 2020.
- [79] Mohammadreza Sadeghi and Narges Armanfard. “Deep Successive Subspace Learning for Data Clustering”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.
- [80] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).
- [81] Trupti M Kodinariya, Prashant R Makwana, et al. “Review on determining number of Cluster in K-Means Clustering”. In: *International Journal* 1.6 (2013), pp. 90–95.

- [82] Glenn W Milligan and Martha C Cooper. “An examination of procedures for determining the number of clusters in a data set”. In: *Psychometrika* 50 (1985), pp. 159–179.