Deep Clustering with Self-supervision using Pairwise Data Similarities

Mohammadreza Sadeghi*, and Narges Armanfard

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 K-dimensional space that is capable of accommodating more complex cluster distributions; hence, providing more accurate clustering performance. 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 through conducting a rigorous set of experiments.

Index Terms—Deep clustering, autoencoder, pairwise data similarity, clustering with soft assignments, cluster-specific loss.

◆

1 INTRODUCTION

In 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 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], and machine vision [4]. 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 [5].

One of the most common approaches in obtaining the lower dimensional space is based on autoencoder (AE) and its variations [6, 7, 8]. An AE consists of two networks: encoder and decoder. Encoder maps the original input space onto a latent space, while 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. [9, 10, 11, 12, 13], 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.

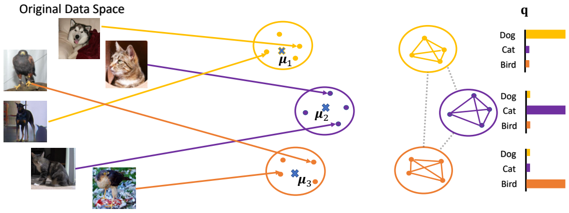


Fig. 1. The motivation of the proposed DCSS method. Arrows show the nonlinear mapping, using the AE, from the original input space to the AE's latent space (i.e the $\mathbf{u}$ space). DCSS employs pairs of similar and dissimilar samples to create the K-dimensional space $\mathbf{q}$ in which pairwise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

All the related existing algorithms, e.g. [9, 10, 14, 15, 16, 17], 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 closest center. However, such crisp cluster assignment of the data in an intermediate training iteration may mislead the training procedure if 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 exist a high uncertainty when deciding

---

- M. Sadeghi and N. Armanfard are with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada.
E-mail: {mohammadreza.sadeghi, narges.armanfard}@ mcgill.ca
- M. Sadeghi and N. Armanfard are also with Mila-Quebec AI Institute, Montreal, QC, Canada.
- * Corresponding author.

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.

Almost all the deep clustering algorithms neglect the important relevant information available in sample pairs while the effectiveness of such information has been proven in the *supervised* learning methods [18, 19, 20, 21] 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 [18, 19]. 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 the 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. Recently, the contrastive leaning concept is employed for unsupervised data clustering [22] where, since the data class labels are unknown, data augmentation is employed to artificially create positive and negative pairs. The positive pairs consist of samples augmented from the same instance, and the negative pairs otherwise. The main drawback of this algorithm is where augmentation of *all* other samples are considered as negative pairs, while indeed some of the samples belong to the same cluster as of the anchor sample. This may mislead the algorithm's training procedure and/or result in a slow convergence.

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 characteristics of the different clusters. The only existing algorithm that explicitly employs cluster-specific losses is presented in [23]. 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 followings: 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 avoids the error propagation issue caused by the uncertain crisp assignments through 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 centres. 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 pair of samples contribute to training MNet. Due to the curse of dimensionality [24], 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 reliable[1] 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 performance of the existing AE-based clustering methods, e.g. [25, 11, 10, 9]. 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 is presented in Section 4. Section 7 conveys the gist of this paper. Mathematical proofs in support of the proposed method are presented in the supplementary material.

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 [26] on non-deep learning based clustering methods. The following focuses on the review of some related deep clustering methods.

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. For instance, [27] encourages predicted representations of the augmented data points to be close to those of the original data points through maximizing the information-theoretic dependency between data and their predicted representations. RUC [28] proposes a two steps 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. [29] 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 [22, 30]. As is discussed in Section 1, such algorithms first need to construct negative and positive pairs by applying

---

[1] In 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.

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 [31].

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 [32, 33], learning the lower representation of the data points is separated from the clustering task. In [32], 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 [34, 35] is a key branch of clustering, which 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. [33] takes advantage of a deep autoencoder to find 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) [25] first trains a stacked autoencoder layer by layer using the reconstruction losses, and then removes the decoder and updates the encoder part through 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 [10, 9, 11], propose to take advantage of the AE's decoder as well as 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) [11] tries to improve 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) [13] 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) [10] 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 into 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 deep k-means (DKM) algorithm [9], which has the same objective function as of DCN, network parameters and cluster centers are updated simultaneously through minimizing its objective function using stochastic gradient descent.

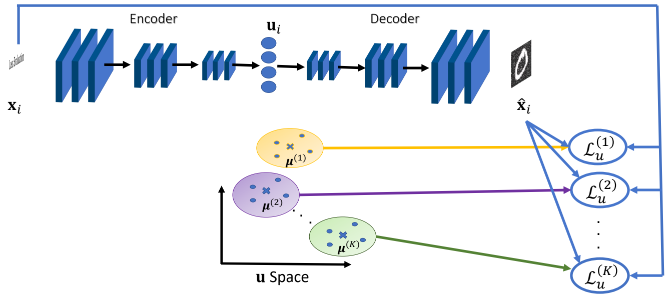
Spectral clustering [36, 37] is a clustering approach that is based on building a graph of data points in the original space and then embedding the graph into a lower-dimensional space in which similar samples sit close to each other. Spectral clustering has been employed in DNN-based methods [12, 38]. For example, deep spectral clustering (DSC) is recently presented in [12]. 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 latent space. DSC considers reconstruction, mutual information, and spectral clustering losses for networks training.

2.1.2 Deep generative based algorithms

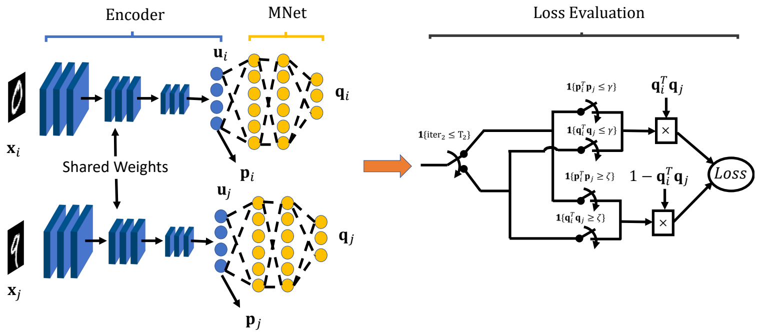
Variational autoencoders (VAEs) [39] and Generative adversarial networks (GANs) [40] are among the most well-known deep generative models which are effective for data clustering. For example, variational deep embedding (VaDE) [41] finds a latent space which captures the data statistical structure that 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 (DAC) [42] is another generative model that applies the adversarial autoencoder [43] to clustering. Adversarial autoencoder employs an adversarial training procedure to match the aggregated posterior of the latent representation with a Gaussian Mixture distribution. DAC's objective function includes a reconstruction term, Gaussian mixture model likelihood, and the adversarial objective. More generative based models can be found in [31]. 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 task [31]. [44] 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. [45] expands the idea of GMM to 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

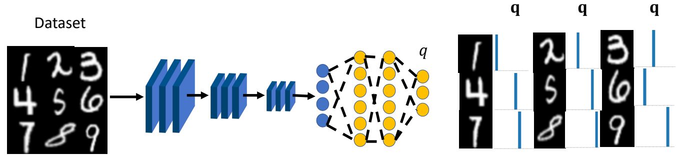
Consider a K-clustering problem which aims to partitioning a given dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$ into K disjoint clusters, where $\mathbf{x}_i$ indicates the ith data sample, N is the number of data points, and K is a predefined user-settable parameter. DCSS utilizes an AE, consisting of an encoder and a decoder network respectively denoted by $f(.)$ and $g(.)$. Latent



(a) The first phase of DCSS


Encoder
MNet
Loss Evaluation
$\mathbf{u}_i$
$\mathbf{q}_i$
$\mathbf{x}_i$
$\mathbf{p}_i$
Shared Weights
$\mathbf{u}_j$
$\mathbf{q}_j$
$\mathbf{x}_j$
$\mathbf{p}_j$
$\mathbb{1}\{\text{iter}_2 \le T_2\}$
$\mathbb{1}\{\mathbf{p}_i^T\mathbf{p}_j \le \gamma\}$
$\mathbb{1}\{\mathbf{q}_i^T\mathbf{q}_j \le \gamma\}$
$\mathbf{q}_i^T\mathbf{q}_j$
$\mathbb{1}\{\mathbf{p}_i^T\mathbf{p}_j \ge \zeta\}$
$1-\mathbf{q}_i^T\mathbf{q}_j$
$\mathbb{1}\{\mathbf{q}_i^T\mathbf{q}_j \ge \zeta\}$
×
×
Loss

(b) The second phase of DCSS



(c) Final cluster assignment

Fig. 2. (a) Training scheme of the first phase of DCSS. (b) Training procedure of the second phase of DCSS; at the outset, when $\text{iter}_2 \le T_2$, 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 **q** space.

representation of X is denoted by $U = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$, 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. Center of the kth 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 kth element indicates the probability of $\mathbf{x}_i$ belonging to the kth data cluster. In this paper, the output of MNet for the ith data point is denoted by $\mathbf{q}_i = M(\mathbf{u}_i; \boldsymbol{\theta}_M)$, where $M(.)$ 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 kth run, the AE focuses on reconstruction and centering of the data points that are more probable to belong to the kth data cluster.

Loss function of the kth 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 1.5 in all experiments.

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

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

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

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

Algorithm 1 Clustering procedure using DCSS

```
Input: Data points $X$, $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$, $\boldsymbol{\theta}_M$, $\boldsymbol{\mu}^{(k)}$ for $k = 1, \ldots, \mathrm{K}$
Output: $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_M$

Phase 1:
Initialize $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ with a pre-trained network (see Section 2 of the supplementary material).
for iter$_1 \in \{1, 2, ..., \mathrm{MaxIter}_1\}$ do
    for $k \in \{1, 2, ..., \mathrm{K}\}$ do
        Compute $p_{ik}$ using (2), for $i \in \mathfrak{B}$
        Update AE's parameters by employing (1a) as loss function
    end for
    Every $T_1$ iterations, update cluster centers using (3)
end for

Phase 2:
for iter$_2 \in \{1, 2, ..., \mathrm{MaxIter}_2\}$ do
    if iter$_2 \leq T_2$ then
        Compute vectors $\mathbf{p}_i$ for $i \in \mathfrak{B}$
        Compute vectors $\mathbf{q}_i$ for $i \in \mathfrak{B}$
        Update $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_M$ to minimize (4)
        Update centers $\boldsymbol{\mu}^{(k)}$, $k = 1, \ldots, \mathrm{K}$ , using (3)
    else
        Compute $\mathbf{q}_i$ for $i \in \mathfrak{B}$
        Update $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_M$ to minimize (5)
    end if
end for

Final Cluster Assignments:
Compute $\boldsymbol{q}_i$ for $\boldsymbol{x}_i$, $i = 1, \ldots \mathrm{N}$
Assign each data sample to the most probable cluster
```

Since data clustering is an unsupervised task, the data cluster memberships are unknown at the problem outset. As such, at the kth 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 kth data cluster, denoted by p_{ik} defined in (2) where $\mathbf{p}_i = [p_{i1}, \ldots, p_{iK}]$. 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 kth run.

Every T_1 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.

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

Block diagram of the first phase is shown in Fig. 2(a). As is demonstrated in our experiments (see Section 4.4), 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 jth neuron, i.e. q_{ij}, denotes the probability of $\mathbf{x}_i$ belonging to the jth 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 kth element of $\mathbf{p}_i$ ($\mathbf{q}_i$) denotes the membership of $\mathbf{x}_i$ to the kth 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 $\mathbb{1}\{.\}$ is the indicator function.

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

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 $\boldsymbol{\theta}_e$ are also updated through back-propagation, in an end-to-end manner. After completing each training epoch, centers $\boldsymbol{\mu}^{(k)}, k = 1, \ldots, \mathrm{K}$, 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 back propagation process when minimizing $\mathcal{L}'_M$.

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

As is proved in Appendix A of the supplementary material file, a proper choice of hyperparameters ζ and γ is respectively $\frac{2}{3} < \zeta$ and $\gamma < \zeta^2$. In our experiments ζ and γ are set to 0.8 and 0.2, respectively. Moreover, Appendix A presents several mathematical proofs that show, under certain assumptions, the final $\mathbf{q}$ vector for a query sample is very close to an one-hot vector where the index of the maximum element of the vector indicates the cluster label. This also shows that similar (dissimilar) samples tend to sit in the same (different) data cluster(s).

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.

4 EXPERIMENTS

In this section, the effectiveness of our proposed DCSS framework is demonstrated on seven benchmark datasets through conducting a rigorous set of experiments. The DCSS clustering performance on these seven benchmark datasets is compared with thirteen clustering methods. The DCSS code is available: https://github.com/Armanfard-Lab/DCSS.

4.1 Datasets

The effectiveness of the proposed method is shown on seven 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 [25, 9, 11, 10, 12]. The datasets are:

(1) MNIST [46] consists of 60,000 training and 10,000 test gray-scale handwritten images with size 28×28. This dataset has 10 classes, i.e. $K = 10$.

(2) Fashion MNIST [47] has the same image size and number of samples as of 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 10 classes of data, i.e. $K = 10$.

(3) 2MNIST is a more challenging dataset created through 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 [48] contains of 9,298 16×16 handwritten images from the USPS postal service. It contains 10 classes of data, i.e. $K = 10$.

(5) CIFAR-10 [49] 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 [50] 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 [49] is similar to the CIFAR-10, except it has 20 super groups based on similarity between images instead of 10 classes. The number of clusters K for this dataset is set to 20.

For RGB image datasets, we apply our proposed DCSS algorithm on the extracted features obtained by a Resnet-152 [51], which is pre-trained on the ImageNet dataset [52]. The network architecture and implementation details of DCSS for each dataset are respectively presented in Section 1 and Section 2 of the supplementary material file.

4.2 Evaluation Metrics

We utilize two standard metrics to evaluate clustering performance, including clustering accuracy (ACC) [53] and normalized mutual information (NMI) [54]. ACC finds the best mapping between the true and predicted cluster labels. NMI finds normalized measure of similarity between two different labels of the same data point. The ACC and NMI formulations are shown below:

$$ACC = \max_{map} \frac{\sum_{i=1}^{N} \mathbb{1}\{l_i = map(c_i)\}}{N} \tag{6a}$$

$$NMI = \frac{I(l;c)}{max\{H(l), H(c)\}} \tag{6b}$$

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

4.3 Clustering Performance

The effectiveness of our proposed DCSS method is compared against thirteen 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 [5], large-scale spectral clustering (LSSC) [55], and locality preserving non-negative matrix factorization (LPMF) [56]. Deep-learning based algorithms are deep embedding clustering (DEC) [25], improved deep embedding clustering (IDEC) [11], deep clustering network (DCN) [10], deep k-means (DKM) [9], variational deep embedding (VaDE) [57], GAN mixture model for clustering (GANMM) [58], deep spectral clustering (DSC) [12], and the very recent contrastive clustering method (CC) [22]. In addition, we report 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 similar architecture as of 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 the first phase of DCSS, presented in Section 3.1, in creating the reliable

TABLE 1
ACC and NMI (in parenthesis) on the benchmark datasets for different clustering methods.

Datasets / Method	MNIST	Fashion MNIST	2MNIST	USPS	CIFAR-10	STL-10	CIFAR-100
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)
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)
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)
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)
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)
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)
GANMM	64.00 (61.00)	34.00 (27.00)	-	50.12 (49.35)	-	-	-
DSC	$97.80^*(94.10^*)$	$66.20^*(64.50^*)$	42.36 (46.53)	$86.90^*(85.70^*)$	22.50 (8.65)	25.60 (15.69)	21.12 (13.00)
AE + k-means	86.03 (80.25)	57.94 (57.15)	44.01 (62.80)	75.11 (74.45)	80.11 (70.35)	95.89 (91.75)	49.86 (48.57)
CC	88.56 (84.21)	64.52 (61.45)	42.15 (58.89)	81.21 (79.45)	79.00 (70.50)	85.00 (76.40)	42.90 (43.10)
DCSS$_u$	95.99 (89.95)	62.90 (63.58)	$45.31^*(63.00^*)$	82.93 (81.84)	$83.40^*(71.32^*)$	$96.02^*(91.90^*)$	$50.30^*(49.80^*)$
DCSS	**98.00 (94.71)**	**66.40 (66.94)**	**48.57 (67.80)**	**87.21 (86.10)**	**85.32 (73.35)**	**97.58 (93.74)**	**51.10 (50.59)**

TABLE 2
ACC and NMI (in parenthesis) with different extracted features.

Dataset	Method	Resnet-34	Resnet-50	Resnet-101	Resnet-152
STL-10	DEC	90.10 (83.40)	91.50 (84.20)	95.30 (90.20*)	95.88 (91.01)
	IDEC	92.70 (84.00)	93.40 (86.15)	95.81* (89.41)	96.00 (92.14)
	DCN	78.12 (77.51)	80.25 (81.12)	90.14 (86.43)	90.25 (84.03)
	DKM	92.40 (88.20*)	91.10 (88.14*)	92.51 (88.57)	95.50 (91.30)
	AE+k-means	92.60 (86.00)	93.20 (87.10)	95.00 (89.90)	95.89 (91.75)
	DCSS$_u$	92.85* (86.26)	93.51* (87.63)	95.20 (90.07)	96.02* (91.90*)
	DCSS	**94.51 (88.35)**	**94.60 (89.37)**	**96.40 (92.00)**	**97.58 (93.74)**
CIFAR-10	DEC	72.51 (64.21)	73.25 (61.10)	78.24 (67.41)	81.10 (68.21)
	IDEC	71.41 (64.01)	74.41 (62.30)	79.65 (68.00)	81.24 (68.57)
	DCN	54.71 (53.29)	56.20 (51.23)	71.00 (46.20)	64.20 (59.02)
	DKM	69.52 (60.41)	70.20 (61.25)	80.31 (68.00)	81.90 (69.10)
	AE+k-means	78.80 (67.81)	75.60 (62.80)	82.90 (70.80)	80.11 (70.35)
	DCSS$_u$	79.02* (68.01*)	76.00* (62.95*)	83.12* (71.20*)	83.40* (71.32*)
	DCSS	**79.80 (70.61)**	**76.40 (64.21)**	**84.50 (73.13)**	**85.32 (73.35)**
CIFAR-100	DEC	41.56 (47.52*)	45.10 (44.25)	43.25 (46.61)	45.38 (49.41)
	IDEC	42.51 (46.41)	45.61 (45.00)	43.45 (47.29)	44.91 (49.25)
	DCN	40.36 (42.58)	41.25 (41.58)	43.15 (42.64)	44.68 (43.00)
	DKM	36.80 (46.82)	35.61 (40.29)	37.84 (47.25)	37.40 (46.20)
	AE+k-means	47.30 (46.10)	47.36 (45.09)	47.35 (47.01)	49.86 (48.57)
	DCSS$_u$	47.66* (46.52)	47.80* (45.44*)	47.43* (47.53*)	50.30* (49.80*)
	DCSS	**50.10 (48.90)**	**48.76 (46.40)**	**48.82 (49.00)**	**51.10 (50.59)**

TABLE 3
ACC and NMI (in parenthesis) on the benchmark datasets when employing DCSS as a general framework to improve state-of-the-art AE based clustering methods.

Datasets / Method	MNIST	Fashion MNIST	2MNIST	USPS	CIFAR-10	STL-10	CIFAR-100
DEC+MNet	89.13 (86.97)	61.25 (56.30)	44.25 (57.35)	77.58 (78.15)	83.40 (69.25)	96.10 (92.00)	48.86(50.11)
IDEC+MNet	90.51 (85.42)	60.12 (57.16)	44.83 (58.00)	76.58 (78.14)	83.95 (69.93)	96.18 (92.89)	47.86 (50.00)
DCN+MNet	87.49 (83.25)	54.23 (58.69)	45.62 (48.24)	76.90 (77.59)	71.23 (62.38)	92.59 (86.23)	47.92 (45.81)
DKM+MNet	88.31 (84.52)	57.23 (56.26)	44.34 (49.50)	77.13 (78.02)	82.26 (69.50)	96.00 (91.88)	40.22 (49.81)

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 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 [59].

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

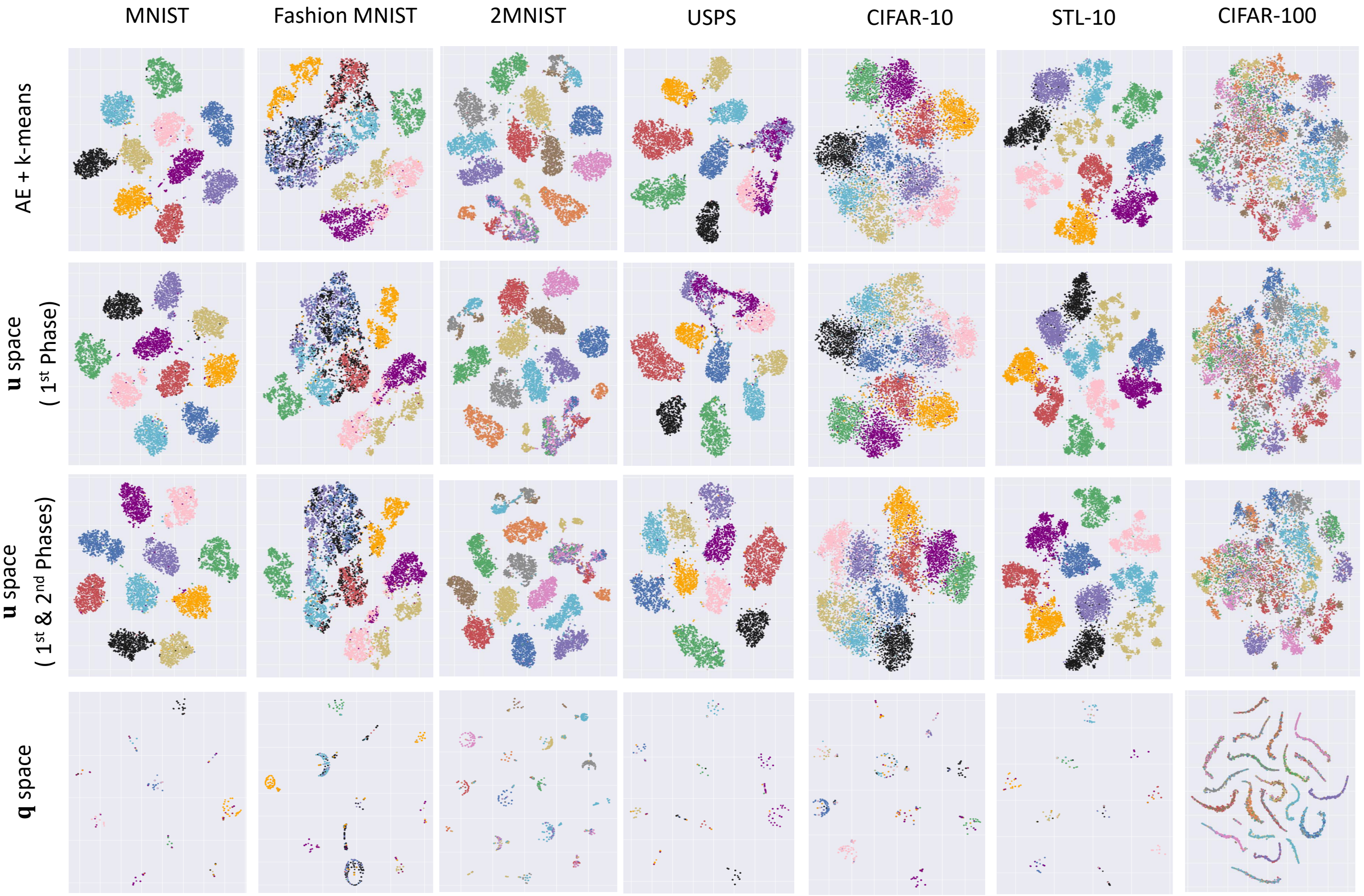


Fig. 3. Clustering visualization of different phases of DCSS using t-SNE, for different benchmark datasets. For reference, the visualization for the baseline model AE+k-means is shown in the first row. Axes range from -100 to 100.

as DCSS$_u$) effectively groups the data points around their corresponding centers. This can be inferred from DCSS$_u$'s ACC and NMI values. Indeed, DCSS$_u$ outperforms all the comparison clustering methods except DSC which is one of the very most recent state-of-the-art AE-based clustering methods. DCSS$_u$ outperforms DSC in 4 out of the 7 datasets and provides competitive results on the remaining three ones. (3) Effectiveness of the self-supervision with similar and dissimilar pair 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) compare 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 t-SNE visualization

Fig. 3 illustrates the effectiveness of different phases of our proposed DCSS framework for all datasets, where t-SNE [60] 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 second row of Fig. 3 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 cluster 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 insufficiency of the reconstruction and centering losses for the clustering task.

The third row of Fig. 3 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 a more clear 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 compare to the corresponding representation shown in the first row.

The last row in Fig. 3 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

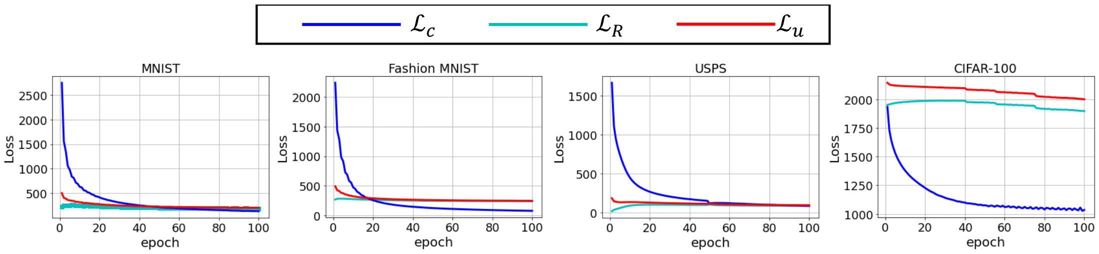


Fig. 4. The reconstruction loss $\mathcal{L}_r$, centering loss $\mathcal{L}_c$, and total loss $\mathcal{L}_u$ of the first phase of DCSS vs. training epochs, for different datasets.

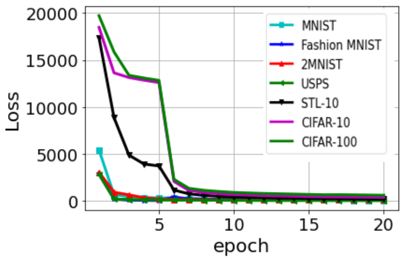


Fig. 5. The second phase's loss function of DCSS for different benchmark datasets.

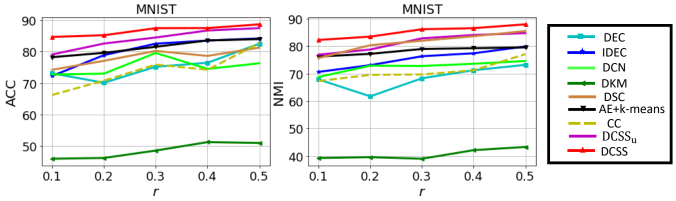


Fig. 6. The clustering performance of different methods on imbalanced samples of MNIST.

clusters of the Fashion MNIST. These clusters are mixed in the $\mathbf{u}$ space but they are completely isolated in the $\mathbf{q}$ space.

For reference, the data representations in the latent space of the baseline model AE+k-means is shown in the first row of Fig. 3. Comparing the data distributions shown in the first and second rows demonstrates the effectiveness of the proposed loss function presented in (1).

4.5 Effect of Pre-trained Network

In this section, we investigate the effect of the structure of the employed pre-trained network for extracting features from RGB images. To this end, we compare the clustering performance of all the deep-learning-based algorithms presented in Section 4.3 using four different ResNet architectures, namely ResNet-34 [51], ResNet-50 [51], ResNet 101 [51] and ResNet-152 [51]. All ResNets are pre-trained on the ImageNet dataset [51]. The corresponding ACC and NMI are reported in Table 2. The best result for each dataset is shown in boldface and the second top results are shown with *. As can be seen, regardless of the employed structure, the proposed DCSS method outperforms all other algorithms on all datasets. In addition, the DCSS$_u$ method is the second top method in 19 out of the 24 reported values.

4.6 Loss function convergence

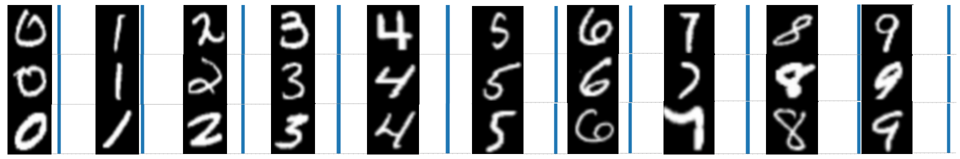
Fig. 4. depicts the average, over different clusters on different batches of data points, of the reconstruction, centering, and total losses corresponding to the first phase of DCSS (i.e. DCSS$_u$) shown in (1). As it can be seen, all losses are converged at the end of training. The noticeable reduction 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 around the centers. Moreover, the figures show that at the first training epochs, our method trades the reconstruction loss for an improved centering performance. This proves insufficiency of the reconstruction loss in creating a reliable latent space for data clustering.

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

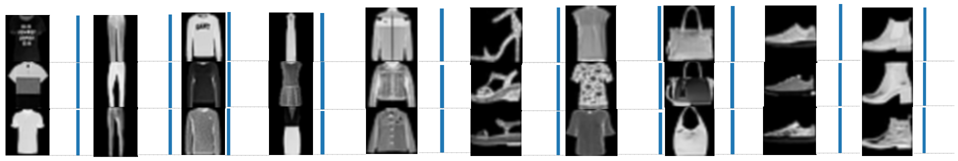
4.7 DCSS as a General Framework

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, algorithms DEC, IDEC, DCN, and DKM 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 3 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 ResNet-152.

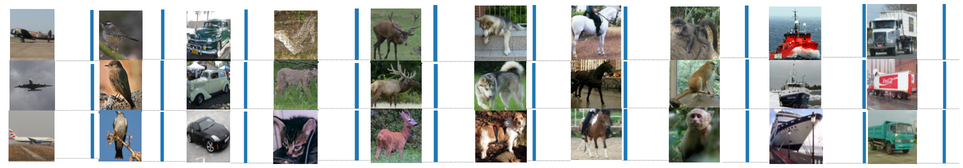
Comparing the clustering results reported in Table 1 and Table 3 confirms the effectiveness of DCSS as a general framework to improve the existing state-of-the-art AE-based clustering methods. On average, MNet improves clustering



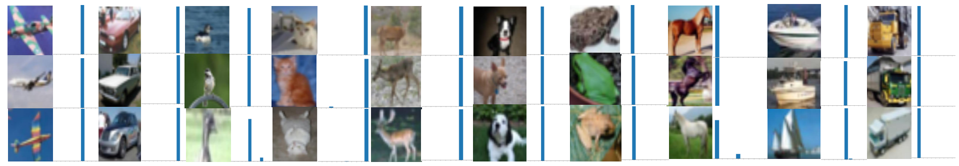
(a) MNIST



(b) Fashion MNIST



(c) STL10



(d) CIFAR10

Fig. 7. Visualization of $\mathbf{q}$ for samples from (a) MNIST, (b) Fashion MNIST, (c) STL-10, and (d) CIFAR-10 datasets. The $\mathbf{q}$ vector for each image is depicted beside the image. The vertical axes range from 0 to 1.

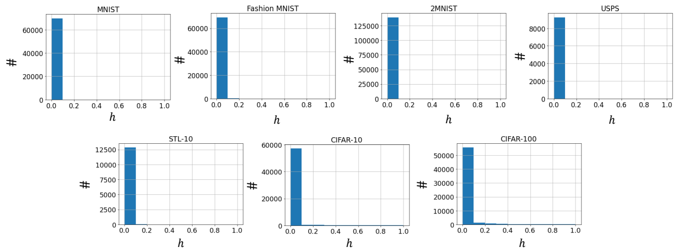


Fig. 8. Histogram plot of $h_i = ||\mathbf{I}_i - \mathbf{q}_i||_1$, $i = 1, \ldots, N$ where $\mathbf{I}_i$ is the one-hot crisp assignment corresponding to $\mathbf{q}_i$.

performance of DEC, IDEC, DCN, and DKM respectively by 3.58% (1.87%), 2.93% (1.54%), 4.04% (2.98%), and 2.56% (1.65%) in terms of ACC (NMI).

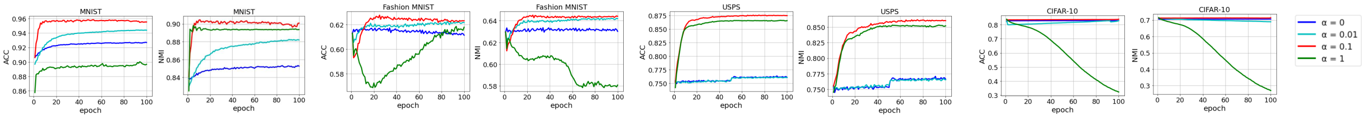
4.8 Performance on Imbalanced Dataset

To demonstrate effectiveness of our proposed DCSS method on 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 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. 6, 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 to the two factors (1) considering an individual loss for every cluster in the 1st phase, and (2) considering the pairwise data relations.

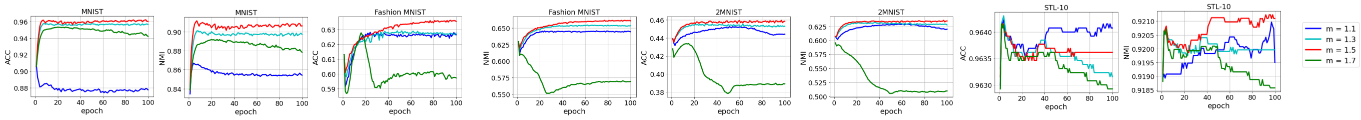
4.9 Visualization of $\mathbf{q}$ vectors

Fig. 7 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 the representations very close to the one-hot vectors. Note that the kth element of $\mathbf{q}_i$ denotes the probability of sample $\mathbf{x}_i$ being in the kth data cluster. The closer $\mathbf{q}_i$ is to the one-hot vector, the more confident crisp cluster assignment can be performed. As is proved in Corollary 1.2 of the supplementary material file, 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.

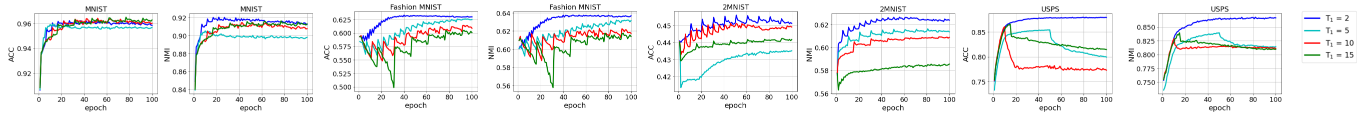
To further demonstrate convergence of the $\mathbf{q}$ representations to one-hot vectors, histogram of residuals $h_i = ||\mathbf{I}_i - \mathbf{q}_i||_1$, $i = 1, \ldots, N$ for all datasets are shown in Fig. 8; where $||.||_1$ indicates the ℓ_1-norm and $\mathbf{I}_i$ is the one-hot crisp assignment 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 element of $\mathbf{q}_i$. As it can be seen in Fig. 8, representation of almost all data points, in the $\mathbf{q}$ space, are very close to their corresponding one-hot vectors.



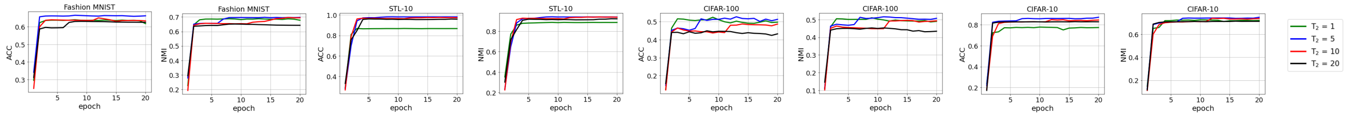
(a) Effect of α on clustering performance



(b) Effect of m on clustering performance

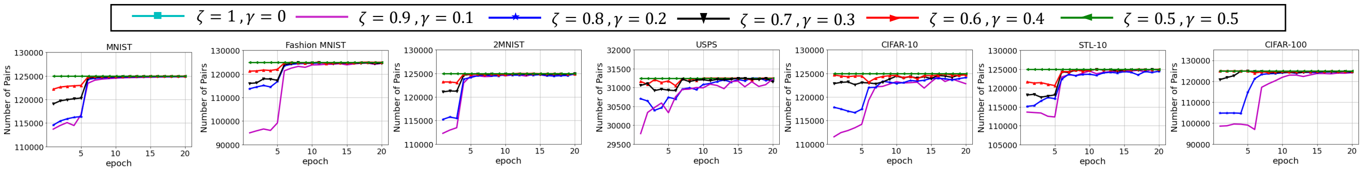


(c) Effect of T_1 on clustering performance

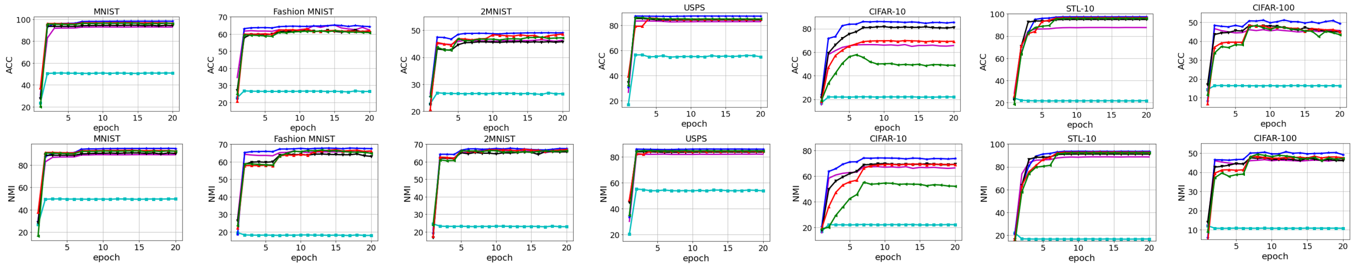


(d) Effect of T_2 on clustering performance

Fig. 9. Sensitivity of DCSS to different hyperparameters.



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



(b) ACC and NMI for different γ and ζ

Fig. 10. 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 value of ζ and γ; the clustering performance of DCSS is less sensitive to the choice of ζ and γ in range of [0.5,0.9] and [0.1,0.5], respectively.

4.10 Hyperparameters Sensitivity

In Fig. 9, 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 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. 9(a), we explore 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, Fashion-MNIST, USPS, and CIFAR-10 datasets. As is shown in this figure, by increasing the value of α from 0 to 0.1, our DCSS performance significantly enhances for most of the datasets 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. In all our experiments, for all datasets, α is set to 0.1.

Fig. 9(b) shows the impact of the level of fuzziness m on clustering performance of DCSS$_u$ for MNIST, Fashion MNIST, 2MNIST, and STL-10 datasets, where $m \in \{1.1, 1.3, 1.5, 1.7\}$. In case of m→ 1 (m→ ∞), group membership vectors converge to one-hot (equal probability) vectors. As it is shown in this figure, as is desired, the DCSS method is not too sensitive to m when it is set to a reasonable value. In all our experiments, for all datasets, m is set to 1.5.

In Fig. 9(c), we scrutinize the effect of update interval

T_1 in 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 smaller value of T_1. In our experiments, for all datasets, T_1 is set to 2.

In Fig. 9(d), we change the number of training epochs T_2, defined in Section 3.2, for Fashion MNIST, STL-10, CIFAR-100, and CIFAR-10 datasets, 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 it is 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 of the $\mathbf{u}$ and $\mathbf{q}$ spaces.

In Fig. 10, 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. 10(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 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. 10(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 DCSS framework is not highly sensitive to the choice of ζ and γ when are set to reasonable values. In all our experiments $\zeta = 0.8$ and $\gamma = 0.2$, for all experiments and datasets.

4.11 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. 11, 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.

5 NETWORKS ARCHITECTURE

The proposed DCSS method includes an autoencoder and a fully connected MNet. This section presents structure of these networks. We use two variations of autoencoders, depending on the dataset nature (i.e. RGB or gray-scale), when training the proposed DCSS framework.

For gray-scale datasets, we propose to use an asymmetric autoencoder; where, following [51], we propose to use the bottleneck layer shown in Fig. 12(c) in the encoder structure. Fig. 12(a) and (b) respectively show the encoder and decoder structures of the proposed asymmetric AE. Employing such asymmetric structure provides a more discriminative latent space. Hyperparameters of the proposed AE for each dataset are indicated in Section 6

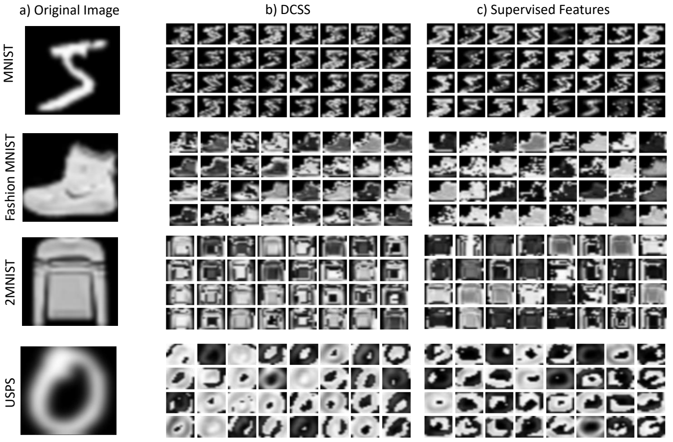


Fig. 11. (a) samples of MNIST, Fashion MNIST, 2MNIST, and USPS. The output of the first convolutional layer using (b) the unsupervised DCSS method, (c) a supervised manner employing the same network structure as of DCSS shown in Section 1 of the supplimentary material file.

For the RGB datasets, we first apply a ResNet-152 [51], pre-trained on ImageNet [52], to extract abstract features. Then we feed the extracted features to a symmetric fully connected AE. Inspired by [25], we set the AE architecture to 2048-500-500-2000-d for RGB datasets, where ReLU activation function is utilized in all layers.

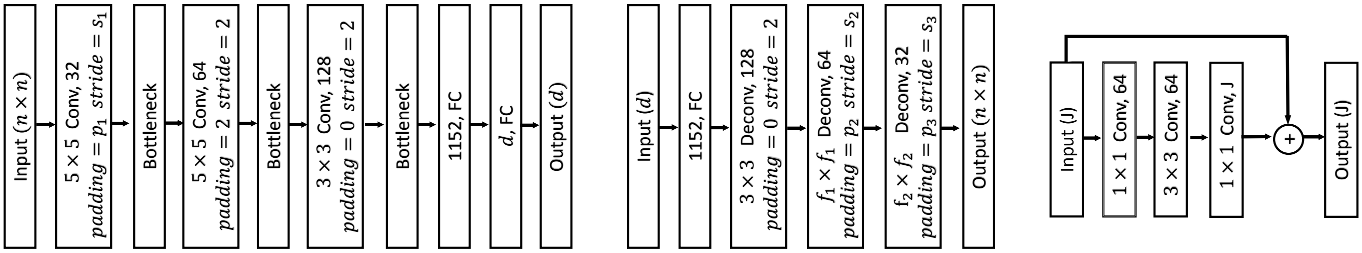
MNet is a fully connected network that takes the d dimensional latent space of the AE ($\mathbf{u}$ space) as input and generates a K dimensional output $\mathbf{q}$. The architecture of MNet is d-128-128-128-K for all datasets except CIFAR-100. Since CIFAR-100 is a more complicated dataset, it needs a more complex MNet architecture; so we set the MNet architecture for CIFAR-100 to d-1000-1000-1000-K. Batch normalization and ReLU activation function are utilized for all datasets in all layers of MNet except the last layer in which we use the soft-max function.

6 IMPLEMENTATION DETAILS

In this section, we discuss hyperparameter values and implementation details of DCSS.

Hyperparameters $n, p_1, s_1, p_2, s_2, p_3, s_3, f_1,$ and f_2 (shown in Fig. 12) are respectively set to 28, 2, 2, 1, 2, 2, 2, 5, and 4 for MNIST, Fashion MNIST, and 2MNIST; these parameters are set to 16, 1, 1, 2, 2, 0, 1, 4, and 5 for the USPS dataset. The latent space dimension d is set to 10 for gray-scale images and 20 for RGB images.

Following [9, 10, 11, 25], in order to initialize $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$ and $\boldsymbol{\mu}^{(k)}$ for $k = 1, \ldots, K$, we train an autoencoder where the end-to-end training is performed by only minimizing the samples reconstruction losses. Adam optimization method [61], with the same parameters mentioned in the original paper are used for training. $\boldsymbol{\theta}_e$, $\boldsymbol{\theta}_d$ are then initialized with the parameters of the trained autoencoder's parameters. We apply k-means algorithm [5] to the latent space of the



(a) Encoder (b) Decoder (c) Bottleneck

Fig. 12. Structure of the proposed asymmetric autoencoder. In the encoder part, in order to obtain an informative lower-dimensional representation of the data points, we propose to use a Bottleneck layer. Following [51], we use the bottleneck layer after 5×5 and 3×3 convolutional layers. The hyperparameter values are presented in Section 6.

trained autoencoder and initialize $\boldsymbol{\mu}^{(k)}, k = 1, \ldots, K$ to the centers defined by k-means.

For all datasets, hyperparameters α, Maxiter$_1$, T$_1$, and m are respectively set to 0.1, 200, 2, and 1.5. The second phase hyperparameters ζ, γ, T$_2$, and MaxIter$_2$ are respectively set to 0.8, 0.2, 5, and 20. We utilize Adam optimizer for updating weights of the AE and MNet and their learning rates are set to 10^{-5} and 10^{-3}, respectively.

All algorithms were implemented in Python using PyTorch framework. All codes are run on Google Colaboratory GPU (Tesla K80) with 12GB RAM.

7 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 take 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.

APPENDIX A

Notation clarification: Representation of the ith sample in the $\mathbf{q}$ space is shown by $\mathbf{q}_i$. The kth element of $\mathbf{q}_i$ is shown by $q_{ik}, k = 1, \ldots, K$, 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, $0 \leq q_{il} \leq 1$ 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 A.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 A.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 ith and jth data points. Then :*

$$\mathbf{q}_i^T \mathbf{q}_j \leq \min\left\{\max_l\{q_{il}\}, \max_l\{q_{jl}\}\right\} \tag{7}$$

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:

$$\mathbf{q}_i^T \mathbf{q}_j \leq \mathbf{q}_i^T \mathbf{q}_j^*, \tag{8}$$

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

$$r = \arg\max_l\{q_{il}\} \tag{9}$$

In the following, we first prove by contradiction that $\mathbf{q}_j^*$ must be a one-hot vector. Then we prove (7).
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:

$$\exists\ e : e \neq r \ \text{ and } \ q_{je}^* \neq 0. \tag{10}$$

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

$$\hat{q}_{jl} = \begin{cases} q_{je}^* + q_{jr}^* & \text{if } l = r \\ 0 & \text{if } l = e \\ q_{jl}^* & O.W \end{cases}, \tag{11}$$

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 (12) and (13), respectively.

$$\mathbf{q}_i^T \mathbf{q}_j^* = q_{ir} q_{jr}^* + q_{ie} q_{je}^* + \sum_{l \neq r,e} q_{il} q_{jl}^* \tag{12}$$

$$\mathbf{q}_i^T \hat{\mathbf{q}}_j = q_{ir}(q_{jr}^* + q_{je}^*) + q_{ie} \times 0 + \sum_{l \neq r,e} q_{il} q_{jl}^* \tag{13}$$

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 (8); thus, $\mathbf{q}_j^*$ must be a one-hot vector. Therefore:

$$\mathbf{q}_i^T \mathbf{q}_j^* \leq \max_l\{q_{il}\} \tag{14}$$

Considering (14) and (8), we have:

$$\mathbf{q}_i^T \mathbf{q}_j \leq \max_l\{q_{il}\}. \tag{15}$$

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

$$\mathbf{q}_i^T \mathbf{q}_j \leq \max_l\{q_{jl}\}. \tag{16}$$

(15) and (16) proves (7).

☐

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 (7), (15), (16) and the adjacency definition provided in Definition A.1. In other words:

$$\zeta \leq \mathbf{q}_i^T \mathbf{q}_j \leq min(q_{ir}, q_{jo}) \quad \rightarrow \quad \begin{cases} \zeta \leq q_{ir} \\ \zeta \leq q_{jo} \end{cases} \tag{17}$$

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 $o = \arg\max_l \{q_{jl}\}$. ☐

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. existence of at least one adjacent (i.e. similar) sample for a data point, on our datasets. Fig. 13 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 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 13. All in all, we can roughly assume that each data point has at least one adjacent sample.

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

$$\begin{cases} \zeta \leq \max_l \{q_{il}\} \\ \zeta \leq \max_l \{q_{jl}\} \end{cases}. \tag{18}$$

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

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:

$$\begin{cases} \zeta \leq q_{i\beta} \\ \zeta \leq q_{k\beta} \end{cases}. \tag{19}$$

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

$$\mathbf{q}_i^T \mathbf{q}_k = q_{i\beta} q_{k\beta} + \sum_{l \neq \beta} q_{il} q_{kl} \tag{20}$$

Therefore, $q_{i\beta} q_{k\beta} \leq \mathbf{q}_i^T \mathbf{q}_k$. Also, we know i and k are dissimilar. From (19), we can conclude that:

$$\zeta^2 \leq q_{i\beta} q_{k\beta} \leq \mathbf{q}_i^T \mathbf{q}_k \leq \gamma \tag{21}$$

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

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 ith and jth 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:

$$\begin{aligned} \mathbf{q}_i^T \mathbf{q}_j &= q_{ir} q_{jr} + q_{io} q_{jo} + \sum_{l \neq o, r} q_{il} q_{jl} \\ \xrightarrow{\forall l:\ q_{jl} \leq q_{jo}} &\leq q_{ir} q_{jr} + q_{io} q_{jo} + \sum_{l \neq o, r} q_{il} q_{jo} \\ &= q_{ir} q_{jr} + q_{jo} \Big(\sum_{l \neq r} q_{il} \Big) \\ \xrightarrow{\sum_{l \neq r} q_{il} = 1 - q_{ir}} &= q_{ir} q_{jr} + q_{jo}(1 - q_{ir}) \\ \xrightarrow{q_{jr} \leq 1 - q_{jo}} &\leq q_{ir}(1 - q_{jo}) + q_{jo}(1 - q_{ir}) \\ \xrightarrow{\text{from (17)}} &\leq q_{ir}(1 - \zeta) + q_{jo}(1 - \zeta) \\ \xrightarrow{\{q_{ir},\ q_{jo}\} \in [0,1]} &\leq 2(1 - \zeta). \end{aligned} \tag{22}$$

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 (22), 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:

$$\begin{aligned} &\zeta \leq \mathbf{q}_i^T \mathbf{q}_j \leq 2(1 - \zeta) \\ \rightarrow &\zeta \leq 2(1 - \zeta) \rightarrow \zeta \leq \frac{2}{3}. \end{aligned} \tag{23}$$

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 $\zeta = 0.8 > \frac{2}{3}$. ☐

Corollary 2.1. *Assume* $\zeta > \frac{2}{3}$. *Consider three data points i, j, and k. If i and j 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. ☐

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.* $\mathbf{q}_j^T \mathbf{q}_k \not< \gamma$).

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 cluster of the corresponding sample, we have:

$$\eta = \arg\max_l \{q_{jl}\} = \arg\max_l \{q_{kl}\}. \tag{24}$$

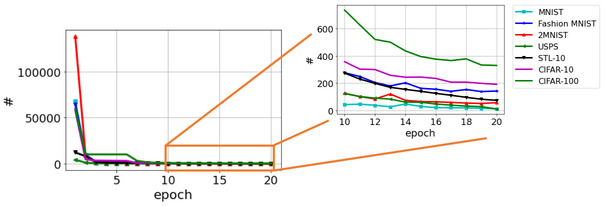


Fig. 13. Number of data points that do not have any adjacent neighbors during the DCSS training in the second phase.

Thus,

$$\mathbf{q}_j^T \mathbf{q}_k = \sum_{l \neq \eta} q_{jl} q_{kl} + q_{j\eta} q_{k\eta}. \tag{25}$$

Therefore,

$$\mathbf{q}_j^T \mathbf{q}_k \geq q_{j\eta} q_{k\eta}. \tag{26}$$

Since $\zeta \leq \mathbf{q}_i^T \mathbf{q}_j$ and $\zeta \leq \mathbf{q}_i^T \mathbf{q}_k$, we can infer (27) from (17).

$$\begin{cases} \zeta \leq q_{j\eta} \\ \zeta \leq q_{k\eta} \end{cases}. \tag{27}$$

From (26) and (27), we can obtain below:

$$\zeta^2 \leq q_{j\eta} q_{k\eta} \leq \mathbf{q}_j^T \mathbf{q}_k. \tag{28}$$

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. $\gamma = 0.2 < 0.8^2$. □

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] 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.

[5] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[6] 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.

[7] Chunfeng Song et al. “Auto-encoder based data clustering”. In: *Iberoamerican congress on pattern recognition*. Springer. 2013, pp. 117–124.

[8] 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.

[9] 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.

[10] Bo Yang et al. “Towards k-means-friendly spaces: Simultaneous deep learning and clustering”. In: *international conference on machine learning*. 2017, pp. 3861–3870.

[11] Xifeng Guo et al. “Improved deep embedded clustering with local structure preservation.” In: *International Joint Conference on Artificial Intelligence(IJCAI)*. 2017, pp. 1753–1759.

[12] 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.

[13] Mohammad Reza Sadeghi and Narges Armanfard. “IDECF: Improved Deep Embedding Clustering With Deep Fuzzy Supervision”. In: *ICIP 2021*. 2021.

[14] Yaniv Opochinsky 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.

[15] 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.

[16] 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.

[17] Jing Pan et al. “Image deep clustering based on local-topology embedding”. In: *Pattern Recognition Letters* 151 (2021), pp. 88–94.

[18] Mahmut Kaya and Hasan Şakir Bilge. “Deep metric learning: A survey”. In: *Symmetry* 11.9 (2019), p. 1066.

[19] Jian Wang et al. “Deep metric learning with angular loss”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2593–2601.

[20] 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.

[21] 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.

[22] Yunfan Li et al. “Contrastive clustering”. In: *2021 AAAI Conference on Artificial Intelligence (AAAI)*. 2021.

[23] Yaniv Opochinsky 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.

[24] 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.

[25] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised deep embedding for clustering analysis”. In: *International conference on machine learning*. 2016, pp. 478–487.

[26] Dongkuan Xu and Yingjie Tian. "A comprehensive survey of clustering algorithms". In: *Annals of Data Science* 2.2 (2015), pp. 165–193.

[27] Weihua Hu et al. "Learning discrete representations via information maximizing self-augmented training". In: *International conference on machine learning*. PMLR. 2017, pp. 1558–1567.

[28] 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.

[29] Jianlong Chang et al. "Deep Self-Evolution Clustering". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2020), pp. 809–823.

[30] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[31] 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.

[32] Peihao Huang et al. "Deep embedding network for clustering". In: *2014 22nd International conference on pattern recognition*. IEEE. 2014, pp. 1532–1537.

[33] Fei Tian et al. "Learning deep representations for graph clustering". In: *28th AAAI Conference on Artificial Intelligence*. 2014.

[34] Satu Elisa Schaeffer. "Graph clustering". In: *Computer science review* 1.1 (2007), pp. 27–64.

[35] 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.

[36] 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.

[37] Santo Fortunato. "Community detection in graphs". In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[38] Uri Shaham et al. "Spectralnet: Spectral clustering using deep neural networks". In: *arXiv preprint arXiv:1801.01587* (2018).

[39] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[40] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).

[41] Zhuxi Jiang et al. "Variational deep embedding: An unsupervised and generative approach to clustering". In: *arXiv preprint arXiv:1611.05148* (2016).

[42] Warith Harchaoui, Pierre-Alexandre Mattei, and Charles Bouveyron. "Deep adversarial Gaussian mixture auto-encoder for clustering". In: (2017), pp. 1–5.

[43] Alireza Makhzani et al. "Adversarial Autoencoders". In: *International Conference on Learning Representations*. 2016. URL: http://arxiv.org/abs/1511.05644.

[44] 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.

[45] Yang Yu and Wen-Ji Zhou. "Mixture of GANs for Clustering." In: *IJCAI*. 2018, pp. 3047–3053.

[46] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[47] 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).

[48] 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.

[49] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[50] 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.

[51] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[52] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. IEEE. 2009, pp. 248–255.

[53] 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.

[54] 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.

[55] Xinlei Chen and Deng Cai. "Large scale spectral clustering with landmark-based representation". In: *25th AAAI conference on artificial intelligence*. 2011.

[56] Deng Cai et al. "Locality preserving nonnegative matrix factorization". In: *Twenty-first international joint conference on artificial intelligence*. 2009.

[57] Zhuxi Jiang et al. "Variational deep embedding: An unsupervised and generative approach to clustering". In: *arXiv preprint arXiv:1611.05148* (2016).

[58] Yang Yu and Wen-Ji Zhou. "Mixture of GANs for Clustering." In: *IJCAI*. 2018, pp. 3047–3053.

[59] 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.

[60] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[61] Diederik P Kingma and Jimmy Ba. "Adam: a method for stochastic optimization". In: *arXiv preprint arXiv: 1412.6980* (2014).

Mohammadreza Sadeghi is currently a Ph.D. Candidate at the Department of Electrical and Computer Engineering, McGill University. He is also affiliated with Mila Quebec AI-Institute, Montreal, QC, Canada. He received his B.Sc. degree in Electrical Engineering from University of Tehran, Tehran, Iran, in 2019. His research interest includes deep learning, machine learning, and deep subspace learning for data clustering.

Narges Armanfard is currently an Assistant Professor (tenure-track) at the Department of Electrical and Computer Engineering, McGill University and Mila Quebec AI-Institute, Montreal, Quebec, Canada. She received her Ph.D. degree in Electrical and Computer Engineering from McMaster University, Hamilton, ON, Canada, in 2016. She completed her postdoctoral studies at the University of Toronto and University Health Network in 2018. She performs fundamental and applied research in machine learning. Her current research interests include machine learning and related areas in computer vision, reinforcement learning, subspace learning for data clustering and classification, and anomaly detection.