Image and Vision Computing 136 (2023) 104712

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Automatic Deep Sparse Multi-Trial Vector-based Differential Evolution clustering with manifold learning and incremental technique

Parham Hadikhani [a,*], Daphne Teck Ching Lai [a], Wee-Hong Ong [a], Mohammad H. Nadimi-Shahraki [b]

[a] School of Digital Science, Universiti Brunei Darussalam, Brunei
[b] Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Iran

ARTICLE INFO

Article history:
Received 5 February 2023
Received in revised form 5 May 2023
Accepted 15 May 2023
Available online 29 May 2023

Keywords:
Unsupervised learning
Deep clustering
Feature extraction
Dimension reduction
Image clustering
Evolutionary algorithm
Differential evolution
Auto-encoder

ABSTRACT

Most deep clustering methods despite utilizing complex networks to learn better from data, use a shallow clustering method. These methods have difficulty in finding good clusters due to the lack of ability to handle between local search and global search to prevent premature convergence. In other words, they do not consider different aspects of the search and it causes them to get stuck in the local optimum. In addition, the majority of existing deep clustering approaches perform clustering with the knowledge of the number of clusters, which is not practical in most real scenarios where such information is not available. To address these problems, this paper presents a novel automatic deep sparse clustering approach based on an evolutionary algorithm called Multi-Trial Vector-based Differential Evolution (MTDE). Sparse auto-encoder is first applied to extract embedded features. Manifold learning is then adopted to obtain representation and extract the spatial structure of features. Afterward, MTDE clustering is performed without prior information on the number of clusters to find the optimal clustering solution. The proposed approach was evaluated on various datasets, including images and time-series. The results demonstrate that the proposed method improved MTDE by 18.94% on average and compared to the most recent deep clustering algorithms, is consistently among the top three in the majority of datasets.

1. Introduction

Clustering is a fundamental technique in the fields of machine learning, pattern recognition, computer vision, and data compression [1]. The purpose of clustering is to find distinct groups in a dataset. Groups are created in such a way that the data points of one group are similar and those of different groups are different. Clustering can be used in a variety of applications including customer segmentation [2,3], image Segmentation [4,5], anomaly detection [6,7], activity discovery [8], recommender systems [9,10] and network analysis [11–13].

Most clustering methods have problems handling large data [1] due to ineffectiveness of the criteria used to find similarities between data points. To address the issue with high dimensional data sets, linear dimension reduction and feature extraction methods such as principal component analysis (PCA) [14], linear discriminant analysis (LDA) [15] and non-linear methods such as Kernel PCA [16] and T-distributed Stochastic Neighbor Embedding (t-SNE) [17] have been used. These methods project data points into a feature space that is easier to classify. However, due to the complex structure of the data, it is still challenging and time-consuming to find useful features that produce good clustering performance [1].

Deep neural networks learn features to represent data effectively without any specialized knowledge of data [18]. Networks such as Auto-Encoder (AE) learns data representation completely unsupervised. An AE consists of two parts, encoder and decoder. In the encoder part, the inputs are compressed and the decoder part tries to regenerate the received input by decoding the compressed representation of the data. The goal here is to reproduce the received input [19], thereby learning the most effective representation of the data.

One problem with a fully connected AE is that in some neurons, after a few epochs, it copies the input as it is, unable to learn and extract essential features [19]. To prevent this, a sparse constraint is used on the activation function of each neuron in the network to disable neurons that have no effect on learning. Another problem of AE is losing the information of distances between features in the representation. This is addressed by applying manifold learning that maintains this information in the representation. Many have proposed combining clustering with deep learning to represent discriminative data features [20–26]. However, most use simple clustering algorithms such as k-means to cluster data points. Also, the majority of deep clustering methods require prior information on the number of clusters which makes them not applicable in many real scenarios. Moreover, conventional

* Corresponding author.
E-mail addresses: 20h8561@ubd.edu.bn (P. Hadikhani), daphne.lai@ubd.edu.bn (D.T.C. Lai), weehong.ong@ubd.edu.bn (W.-H. Ong).

https://doi.org/10.1016/j.imavis.2023.104712
0262-8856/

clustering techniques suffer from getting stuck in local optima, early convergence, low accuracy, and time-consuming [1]. In addition, various techniques have been proposed to estimate the number of clusters. However, these methods are time-consuming and have low accuracy. In this paper, we propose a novel Automatic Deep Sparse clustering technique based on an evolutionary algorithm called Multi-Trial vector-based Differential Evolution (ADSMTDE) to address the aforementioned drawbacks. This algorithm has a high ability to reach the optimal solution due to the use of different search strategies. These strategies maintain a balance between exploration and exploitation in search space and avoid getting stuck in local optimization and early convergence. An incremental technique is also applied to perform clustering without requiring information about the number of clusters.

The main contributions can be summarized as follows:

- To the best of our knowledge, this is the first deep clustering method that uses an evolutionary algorithm. We propose an end-to-end deep clustering based on the MTDE algorithm that benefits from different search strategies and the appropriate distribution of populations between strategies to find the final solution.
- To improve the extracted features, we apply sparse constraint to AE to minimize redundant information.
- A manifold learning is adopted in AE to preserve distances between extracted features and maintain important ones.
- Incremental technique is employed to perform clustering automatically without requiring prior information about the number of clusters.
- Extensive experiments on seven challenging datasets have shown the benefits of ADSMTDE over a wide range of state-of-the-art deep clustering techniques.

In the rest of this paper, the related works on clustering are reviewed in Section 2. The proposed method is introduced in Sections 3 and 4. The experiments and results are shown in Section 5. Finally, the conclusion is given in Section 6.

2. Related works

2.1. Evolutionary approaches

Classical clustering can be categorized as hierarchical [27], distribution-based [28], density-based [29], learning network clustering [30], and partition clustering [31]. Since clustering is an NP-hard problem, a vast majority of evolutionary clustering methods have been proposed to solve clustering due to their strength in optimization [32]. Hadikhani et al. [33] proposed a hybrid particle swarm optimization with k-means. They applied a Gaussian mutation to produce diverse solutions. Lei et al. [34] designed a fuzzy clustering for image segmentation. They adopted morphological reconstruction and membership filtering in fuzzy clustering for robustness in dealing with noise and increasing clustering speed. In [35], a novel framework utilized an adaptive evolutionary clustering to eliminate the faulty and uninteresting pairings in analysis and processing of information which reduced computation time. Li et al. [36] explored the segmentation problem on texture image and proposed an evolutionary algorithm based on quantum theory. Although some of these works show good performance in small datasets, they performed poorly on high-dimensional data because a suitable data representation for clustering is not prepared [1].

The MTDE algorithm [37], which stands for Memory-based Tournament Differential Evolution, is a variant of the Differential Evolution (DE) algorithm [38] that uses historical data to search for the best performing region within the solution space. In traditional DE, a population of candidate solutions is evolved over multiple generations to find the best solution. However, in MTDE, the historical data from previous generations is also used to guide the search towards the best performing region, resulting in faster convergence and better performance. The MTDE algorithm also employs three search strategies to balance exploration and exploitation in the search space.

2.2. Deep learning approaches

Deep clustering combines representation learning with clustering algorithms. Yang et al. [21] introduced a combination of deep learning with k-means to enhance the clustering performance. The goal of their work was to map data representation to latent space for reducing the dimension and finding the distinct features.

Guo et al. [39] presented a deep clustering that jointly clusters data and learns embedded features by considering simultaneously the clustering loss in the latent layer to disperse the embedded features and the reconstruction loss to ensure that the data's local structure is preserved in embedded space. They also utilized KL divergence loss to enhance the produced features. Menapace et al. [40] proposed a deep clustering method to solve domain adaptation settings. They utilized data from several unlabeled source domains and use a semantic predictor that cluster data from multiple unlabeled source domains. Mutual information maximization and batch normalization were utilized to obtain stable features. Van et al. [41] suggested a two-step approach to cluster data. First, they extracted meaningful features in a self-supervised manner. Afterward, a learnable clustering approach was employed to group the data. Astorga et al. [42] used GAN for learning from data and applied Matching Priors and Conditionals to cluster data. Zhao et al. [43] disentangled the data representation into category part and style part by using augmentation-invariant loss. In addition, prior probability was employed to ensure degenerate solutions are avoided. Niu et al. [44] proposed a clustering approach with a Gaussian attention layer to learn discriminative features. Huang et al. [45] developed a deep clustering algorithm based on partition confidence maximization. They applied partition uncertainty index to assess the overall confidence in the clustering solution. Moreover, they introduced a transformation of the partition uncertainty index in order to facilitate the formulation of the deep learning loss function. Li et al. [46] introduced a novel deep clustering approach based on specified contrastive learning, where contrastive learning is conducted both at the instance-level and cluster-level to determine suitable representations for clustering. Sadeghi et al. [47] introduced a new contrastive loss function to learn a more distinctive representation space while employing instance-level representation to find related samples. They used a learned latent representation to find similarities between samples and trained their network to bring comparable cases that improve the clustering. Mcconville et al. [48] investigated different manifold learning to apply to AE for preserving the local structure of data and also proposed a new way for deep clustering.

In the case of combining subspaces clustering with deep learning, methods such as using k-subspace in deep clustering to reduce the possibility of reconstruction error [49], applying a self-expressive layer between the encoder and the decoder in AE to make use of the self-expressiveness features of subspaces [50], using linear time and space complexity to overcome large-scale subspace clustering [51], and mapping data into the eigenspace by employing spectral clustering as part of a deep learning methodology [52] have been proposed. Although these methods provide a good network to find the hidden structure of data, their problem lies with the use of shallow clustering methods like k-means that easily get stuck in the local optimization or subspace clustering which does not have a specific strategy for dealing with different situations.

This paper adopts sparse and manifold learning techniques to extract better features. These techniques have been used in various machine learning and data analysis applications. However, the way these techniques are incorporated into the deep clustering problem is unique and innovative. In traditional deep clustering methods, features are typically extracted from the input data using pre-trained neural networks. However, this approach can result in insufficient feature extraction,

leading to suboptimal clustering results. To address this limitation, the ADSMTDE method adopts sparse and manifold learning techniques to extract better features that capture the underlying structure of the data. L1 regularization encourages the feature extraction process to produce sparse representations of the input data. This can help to reduce the dimensionality of the feature space and remove noisy or irrelevant features, resulting in more informative features that are better suited for clustering. On the other hand, UMAP aims to capture the underlying structure of the data by mapping the input data to a lower-dimensional manifold. This can help to overcome the curse of dimensionality and improve the quality of the feature representations, making them more suitable for clustering.

2.3. Determining number of clusters

One important problem in clustering is determining the number of clusters. Many methods have been proposed using cluster validity methods to estimate the number of clusters, but these methods are time-consuming and perform inconsistently in estimating the number of clusters. A deep clustering method without knowing the number of clusters was proposed by [53]. They used a recurrent neural network to perform clustering and applied a covariance matrix to estimate the number of clusters.

Different from previous approaches, our proposed method uses sparse constraint in AE to improve the performance of AE in learning the representation of data and manifold learning is applied to preserve the local structure of the data. In the clustering part, our proposed method uses an evolutionary algorithm that performs well in exploration and exploitation. Unlike the use of covariance matrix in [53], which is sensitive to the volatility among data points and makes inaccurate estimation in the presence of outliers, an incremental technique is used to find the number of clusters. In this technique, unlike the common methods that try to estimate the number of clusters, it looks for suitable clusters through an iterative process. This technique is adopted to dynamically update the clustering results as new data points are added to the dataset over time. This allows the clustering algorithm to adapt to changing conditions and to avoid reprocessing the entire dataset each time new data is added, as detailed in Section 4.3.

This is a significant innovation because many existing clustering methods require prior knowledge of the number of clusters, which can be difficult to determine in practice. The proposed method avoids this limitation and provides a fully unsupervised clustering framework that can be applied to growing datasets in real-time.

3. Automatic Deep Sparse clustering technique based on an evolutionary algorithm called Multi-Trial Vector-based Differential Evolution (ADSMTDE)

We propose a fully unsupervised deep clustering. Our method has two novelties: one is the high-quality and small-sized extracted feature set and the other is providing a completely unsupervised clustering framework that does not require any prior information of the number of clusters.

We are not using supervised methods since they require labels for the training. Data labeling is a manual, tedious, and expensive process, and there are no labels in growing data. Consequently, supervised methods are impractical to use in real scenarios. Also, they are not scalable because these methods have been trained based on a specific condition and need to be trained again for new conditions.

Thus, the main motivation of this paper is to propose a method that can group images in real-time and automatically in growing data.

To present an efficient clustering framework, we develop two algorithms. In the first algorithm, we extract the features and in the second, images are automatically clustered.

First, we present a method that can extract high-quality features in an unsupervised way which we consider two important aspects: 1) the extracted features should represent salient characteristics of images, 2) the feature dimension should be small to facilitate the clustering task (a larger dimension confuses the clustering process [54]). To the best of our knowledge, the existing algorithms do not perfectly consider these two aspects and this research addresses this problem.

To cover the first aspect, contrary to N2D [48] which only uses a simple auto-encoder, we add a sparsity constraint into the auto-encoder to exploit and extract more representative and essential features for the input. This is done because without the sparsity constraint an autoencoder would have too many degrees of freedom and then would be very prone to overfitting. Thus, this regularization tries to prevent the overfitting of the model.

Moreover, in a simple auto-encoder, some nodes will copy the input data exactly to the network's output, which creates redundancy in the extracted features and has a detrimental effect on clustering performance. We prevent this problem and extract effective features by using sparsity constraint. We employ a manifold learning method to address the second aspect, reduce dimensionality, and make the extracted feature more clusterable while high-importance features are kept.

We present an automatic clustering based on the MTDE and Incremental Technique (IT) combination in the second algorithm. In N2D, k-means and GMM are used for clustering, but the disadvantage of K-means clustering with different data representations is different results and it can easily get stuck into the local optimum. The drawback of GMM is that there are many parameters to fit, and it usually requires a lot of data and many iterations to get good results. However, the MTDE has better intelligence than K-means and GMM because it uses historical data to search for the best performing region within the solution space.

In addition, MTDE has three search strategies for finding the best answer. These strategies maintain a balance between exploration and exploitation in the search space. Therefore, our method has a much lower possibility of falling into the local optimum than N2D. The other novelty of this research is that the method discovers the number of clusters using IT while state-of-the-art methods such as N2D require knowledge of the number of clusters in advance.

The challenge of finding the number of clusters is that many methods use a cluster validity index to estimate the number of clusters, but they are very time-consuming and inaccurate because selected clusters are not evaluated. We solve this problem by applying MTDE with IT during clustering. Thus, our method can discover high-quality clusters automatically in a time-efficient process by checking the quality of the selected cluster. To the best of our knowledge, this is the first deep clustering method done from beginning to end without any previous knowledge and is completely unsupervised.

3.1. Sparse auto-encoder along with manifold learning

An AE is a three stage neural network that receives its inputs in the first stage, sending these inputs through the hidden layer, and rebuilding them again in its output stage [19]. Training an AE consists of two phases: encoding and decoding. In the first phase, the AE receives its input in the form of a d-dimension tensor x and encodes it into a d'-dimensional tensor y. The encoding of the input data is formulated as follow:

$$y = f(w \times x + b), x \in R^{d} \text{ and } y \in R^{d'} \quad (1)$$

f is the activation function, where the ReLU function is used in this paper. w and b are the weight and bias of the encoder network respectively.

In the next phase, the encoded data (y) is converted to x' in the output layer, which rebuilds the x input. This phase is called decoding as formulated in Eq. (2).

$$x' = f\left(w' \times y + b'\right), y \in R^{d'} \text{ and } x' \in R^{d} \quad (2)$$

w' and b' are the weight and bias of the decoder network, respectively.

To minimize the reconstruction error (distance between x and x' tensors) and optimize the AE parameters, AE is trained with mean square error (MSE) loss function in the Eq. (3) to determine the values of w and b of the network. These parameters are found by minimizing the reconstruction error (distance between x and x' tensors).

$$LossMSE(x, x') = \|x' - x\|^2 \quad (3)$$

After a few epochs, the AE merely replicate the input to the output [19]. To prevent this so that important features are learned, $L1$ Regularization, as sparse constraint is added to the activation function of the hidden neurons. This result in the weight of some neurons that are not effective becoming zero and the extracted features being more robust to noise [55]. By adding $L1 = \lambda * \sum |w_i|$ where w is the activation weights and λ is the regularization parameter, the loss function and the final cost is calculated by Eq. (4).

$$Cost = LossMSE + L1 \quad (4)$$

Although the AE performs well to obtain latent features, it suffers from maintaining the distances within data points in the features. To address this problem and improve the quality of features, a manifold learning technique called UMAP is applied to determine the nearest neighbor points from embedded features using a k-neighbour based graph method. UMAP first creates a weighted k-neighbor graph and then computes a low dimensional layout from this graph. Based on cross-entropy, this low-dimensional arrangement is tuned to have a fuzzy topological representation as close to the original features as feasible. There are various manifold learning techniques such as Isomap and t-SNE. However, their performance in large datasets drop sharply [48]. Recently, Uniform Manifold Approximation and Projection (UMAP) has been introduced by McInnes et al. [56]. This method not only increases the speed of clustering but also performs much better in maintaining the hidden data structure compared to other methods such as t-SNE [56].

4. MTDE clustering

MTDE is an improved DE algorithm [37]. In DE, each member of a swarm (called a vector) represents a potential solution. To produce new solutions in DE, a trial vector is used. The trial vector is obtained from crossover and mutation of two random vectors. The performance of DE strongly depends on how the trial vector is calculated. As it does not examine other dimensions of search and does not establish any balance between global and local search, DE encounters problems in high-volume data [37].

To solve this, MTDE uses three strategies: Representative-based Trial Vector Producer (R-TVP), Local random-based Trial Vector Producer (L-TVP), and Global best History-based Trial Vector Producer (G-TVP). R-TVP enhances the diversity of solutions, L-TVP provides a proper balance of exploration and exploitation as well as rapid convergence, and G-TVP helps to escape from local optima.

MTDE also stores previous inferior solutions that have information about visited places in a repository called lifetime archive to transfer the experience to the next generation. This repository is used by the trial vector to generate diverse solutions and prevent premature convergence. The size of the repository is equal to the size of the population of vectors, and new inferior solutions replace the old ones if the repository is full.

4.1. Population distribution and search strategies

MTDE uses a population distribution mechanism called "winner-based distributing" to balance between exploration and exploitation. This mechanism divides the population between strategies such that a strategy that performed better has a larger population than the rest to increase its impact on the solutions. Initially, due to a lack of evaluation in the strategies' performance, one is randomly selected as the best strategy and in later iterations, the strategy with the highest improved rate ($IR = \frac{IV_x}{N_x}$) value is selected as the best strategy. N_x is the number of population assigned to strategy x and IV_x is the number of improved vectors by strategy x. After selecting the best strategy in the current iteration, vectors are distributed into the strategies randomly based on N_x such that if the population of R-TVP or L-TVP have the highest IR_x, the population of superior strategy (N_{win}) is calculated based on $N_{win} = 0.6 \times N$ and the population of other two strategies are calculated based on $N_{lose} = 0.2 \times N$ where N is the total number of vectors. Otherwise, if G-TVP have the highest IR_x, $N_{win} = 0.2 \times N$ and $N_{lose} = 0.4 \times N$.

In R-TVP strategy, the target vector (x_i) first mutates ($v_{i_{R-TVP}}$) with a random vector in lifetime archive x_{life} and two vectors $x_{i_{best}}$ and $x_{i_{worst}}$ that have the best and worst fitness values among the N_{R-TVP} based on Eq. (5). The trial vector ($u_{i_{R-TVP}}$) is then obtained by calculating the crossover of two transformation matrices M and its binary inverse $\overline{M}$ with the target vector and the mutant target vector, respectively. M with dimensions $N \times D$ is generated by reproducing the square matrix with dimensions $D \times D$ $\frac{N}{D}$ times in $M_{N \times D}$, which is a lower triangular matrix with values of one. The remaining $M_{N \times D}$ rows are filled with the first rows of the square matrix. Rows of $M_{N \times D}$ are then permuted randomly. $\overline{M}$ is obtained by substituting the inverse Boolean value of each element in M.

$$v_{i_{R-TVP}} = x_i + f_i \times \left(x_{i_{best}} - x_i\right) + f_i \times \left(x_{i_{worst}} - x_i\right) + \alpha_1 \times \left(x_{life} - x_i\right) \quad (5)$$

$$u_{i_{R-TVP}} = \left(x_i \times M + v_i \times \overline{M}\right) \quad (6)$$

$$\alpha_1 = 2 - iter \times \left(\frac{2}{MaxIter}\right) \quad (7)$$

Where f is a scaling factor derived from the Cauchy distribution [57], and α_1 is a coefficent derived from Eq. (7).

The global best vector x_{gb} is modified in the G-TVP strategy using two random vectors from the N_{G-TVP} (population of G-TVP). In this strategy, the trial vector ($u_{i_{G-TVP}}$) is created in the same way as R-TVP is created in Eq. (9).

$$v_{i_{G-TVP}} = x_{gb} + \alpha_2 \times \left(x_{r1} - x_{r2}\right) \quad (8)$$

$$u_{i_{G-TVP}} = \left(x_i \times M + v_i \times \overline{M}\right) \quad (9)$$

Unlike the other two strategies, the trial vector ($u_{i_{L-TVP}}$) in the L-TVP approach is produced by individual learning rather than evolution. Therefore, there is no requirement for crossover, and it is computed as follows:

$$u_{i_{L-TVP}} = x_i + f_i \times \left(x_{r1} - x_{r2}\right) + \alpha_2 \times \left(x_{life} - x_i\right) \quad (10)$$

$$\alpha_2 = (initial - final) \times \left(\frac{MaxIter - iter}{MaxIter}\right)^{\mu} \quad (11)$$

where x_{r1} and x_{r2} are two random vectors from the population of L-TVP, x_{life} is a random vector from the lifetime archive, and α_2 is the coefficient obtained by Eq. (11). Furthermore, *initial* and *final* are the initial and final values of the user-defined control parameter α_2, and μ is the vector dimension.

The population is updated in the selection phase by comparing the trial vectors produced in each strategy to their corresponding target

vectors. The target vector is replaced by the trial vector if the trial vector has a higher fitness value, and the target vector is saved in the lifetime archive. Finally, the global vector x_{gb} is chosen as the final optimal solution after reaching the termination condition.

Algorithm 1: ADSMTDE Cluster Algorithm

Input: Dataset=$\{y_1, y_2, \ldots, y_n\}$, $k_{max} = \sqrt{\frac{n}{2}}$, number of epoch ($P$), Batch size ($B$), learning rate ($L$), and sparse rate ($\lambda$), Maxiter=200
Output: k clusters

Initialize the AE with random weights
for *epoch from 1 to P* **do**
  for $i \in 1, \ldots, dataset/B$ **do**
    $features_i$= Latent features of i
    compute loss function (Eq. (4))
Apply UMAP on encoded features
Randomly initialize the population of vectors
for *iter=1 to the Maxiter* **do**
  Distribute vectors using winner-based distributing
  for *each strategy x* **do**
    for x_i *in* N_x **do**
      producing trial vectors with Eq. (6), (9) and (10)
      Evaluate x_i using Eq. (12) and update x_i and x_{life}
      Assign y_i to the cluster with the nearest centroid
  Choose the global best vector (x_{gb})
  //Incremental technique
  for *each cluster of* x_{gb} **do**
    Compute the fitness values of each cluster based on the following equation: $error = \sum_{\forall y_i \in C_j} \|y_i - C_j\|^2$
  Select the cluster with the lowest value of $error$ as c_i^*
  if *member of* $c_i^* \prec Minpt$ **then**
    Add c_i^* to cluster set C
    Remove the selected cluster from dataset by $n^* = Dataset - c_i^*$
    Recalculate the number of cluster by $k = \sqrt{\frac{n^*}{2}}$
  else
    Recalculate the number of cluster by $k = k - 1$
Return C

4.2. Clustering

In clustering, each vector x_i in MTDE represents k cluster centroids, and the structure of each x_i is $(C_{i1}, \ldots, C_{ij}, \cdots, C_{ik})$. Where C_{ij} is the *jth* cluster centroid in the *ith* MTDE vector (x_i). A fitness function computed from Sum of Square Error (SSE) is utilized to evaluate the clustering quality of each vector. SSE is a statistical measure that determines the total deviation of the data points inside a cluster. It estimates the variance inside a cluster by calculating the difference between each data point and its cluster means [58]. The following equation is used to compute SSE.

$$SSE = \sum_{j=1}^{k} \sum_{\forall y_i \in C_j} \|y_i - C_j\|^2 \tag{12}$$

Where the data points y_i belong to the cluster C_j. Each vector is assessed based on SSE at each iteration of clustering and the lowest value being selected. In addition, cluster centroids in each vector are updated by one of the three strategies.

4.3. Incremental technique for determining number of clusters

One important problem in clustering is determining the number of clusters. Most existing solutions are time-consuming and have low accuracy. In this paper, instead of clustering all the data points at once, high-quality clusters are individually identified over time. This solves the problem of finding the number of clusters and greatly increases the clustering speed. After receiving the dataset, the minimum number of data points ($MinPt = \frac{n}{k_{max}}$) in each cluster is first calculated based on the maximum number of clusters (k_{max}) and the number of data points (n) to ensure the clusters have sufficient observations. k_{max} is set to $\sqrt{\frac{n}{2}}$ to choose an appropriate value that shortens computation time. For each k, clustering is performed and a cluster with the lowest SSE is selected. If the number of data points in the selected cluster is greater than or equal to MinPt, the cluster is selected as one of the final clusters and its members are removed from the dataset. A new value of k is then calculated based on $\sqrt{\frac{n^*}{2}}$ where n^* is the pruned population of the dataset. But if the number of data points in the selected cluster is less than MinPt, that cluster is ignored and the value of k is decremented. This process is repeated until k reaches 2. Finally, the found clusters are aggregated and considered as the final solution. The proposed method is shown in Fig. 1 and its pseudocode is shown in 1.

5. Experiments

The proposed method is evaluated based on metrics accuracy (AC) [33] and Normalized Mutual Information (NMI) compared with the latest State-Of-The-Art (SOTA) methods. These methods include 13 recent deep and five conventional and well-known clustering algorithms that they know about the number of clusters.

We compared our proposed method to several baseline approaches, including k-means [59], Subspace Clustering (SC) [60], and Gaussian Mixture Models (GMM) [61]. We also included DE and MTDE as comparison algorithms because our proposed method builds upon these approaches.

Deep clustering methods were compared with our method to show how successful the proposed method is in improving recent methods. Moreover, we compared our method with automatic methods, including AMTDE, DPSO [62], PSOAC [63], KM [64] and DBSCAN [29] to show the effectiveness of the incremental technique.

Experiments were performed on seven datasets including two handwritten digits, the MNIST [65] with 70,000 images of 28×28 pixels and the USPS with 9298 images of 16×16 pixels of 10 different classes of numbers, a challenging Fashion dataset consisting 70,000 clothing images of 10 different classes and two time-series datasets, the Pendigits, which consists of 10,992 sampled points from 10 different numbers that were pressed on a pressure-sensitive tablet, HAR, which consists of recording of 6 activities performed by 30 people using a mobile phone and CIFAR-10 [66], which consists of 60000 32×32 color images in 10 classes, with 6000 images per class and Tiny ImageNet [67] with 100,000 64×64 colored images contains 200 classes and each class has 500 images.

The reported results were obtained from an average of 30 runs. The ground truth is only for evaluating the model provided by the method and not for model selection. For model selection, no prior or labelled information is used.

5.1. Experimental settings

The architecture and layer dimensions of AE were inspired by [48]. The dimensions of the encoder layers are as follows; *input dimension*, 500, 500, 2000, 10 while the dimensions of the decoder layers are in reverse order of the encoder. *Relu* was used as the activation function in all layers and *Adam* method was used to optimize the weights. Also, the number of *epochs* for training the AE was set to 1000. In UMAP, several parameters affect performance, including the number of neighbors used to capture local structure and the feature dimension. For this work, we set the number of neighbors to 10 and the feature dimension to 2. The parameters in the MTDE clustering were set as follows: $Maxiter = 200$, $initial = 0.001$, $final = 2$, and number of vectors $= 200$.

5.2. Complexity

To use the algorithm in a real application, it is important how fast the algorithm runs and how much memory it needs. Such predictions lead

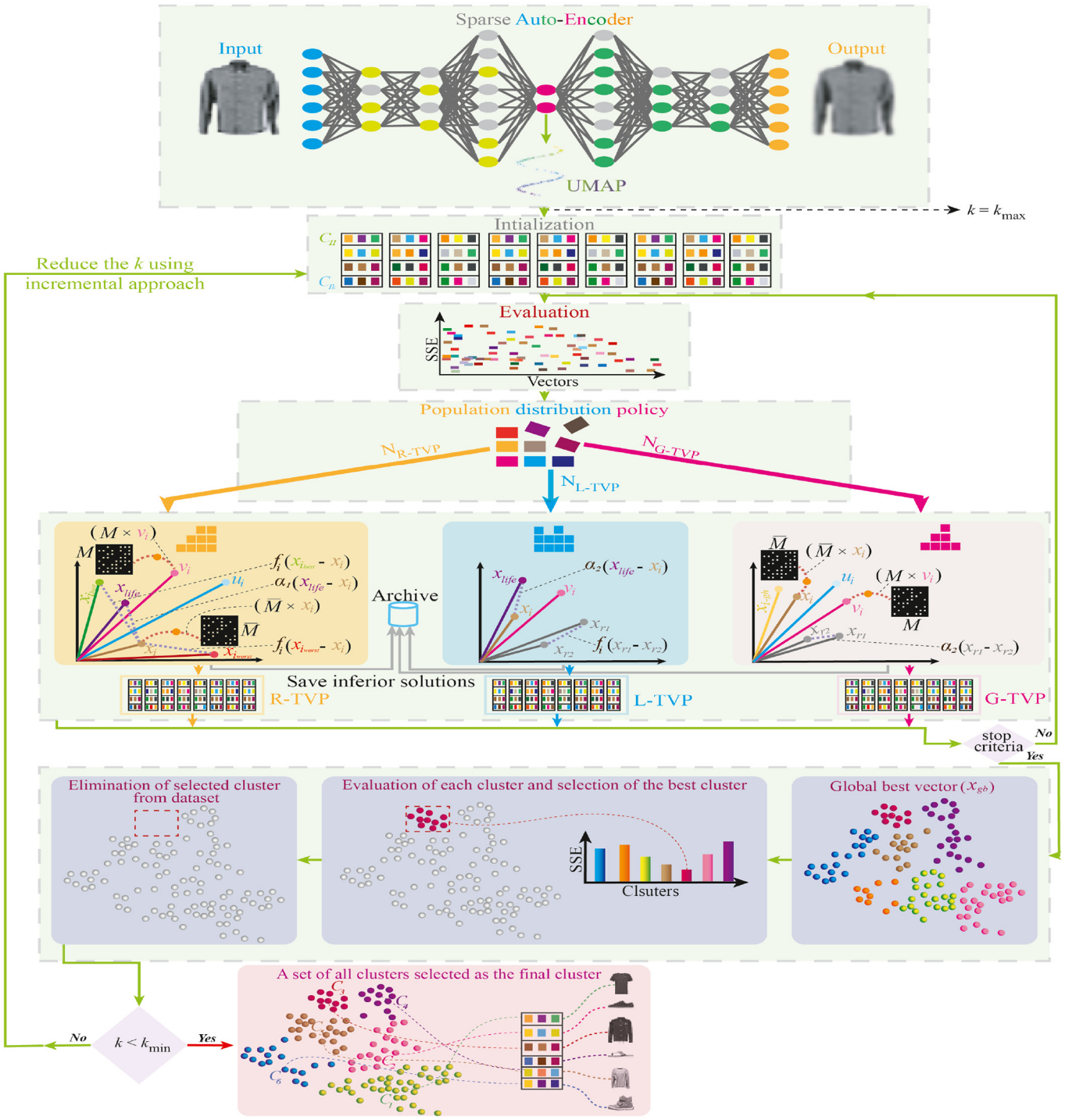


Fig. 1. Illustration of the proposed clustering: After receiving input, features are extracted from the latent layer and UMAP is applied to them to preserve the local structure of data. Until the number of clusters reaches k_{min}, solutions (vectors) are initialized and distributed into the three strategies to optimize them in different ways. Inferior solutions are kept in a repository for sharing information with the next generations. For each k value, a cluster of the best solution (x_{gb}) is selected based on the SSE value if it satisfies *MinPt*. Then, it is removed from the dataset as one of the final clusters. The new number of clusters is then calculated for the size of the pruned dataset.

to the implementation and propose an optimized and suitable algorithm that in addition to increasing the accuracy of the algorithm, it makes the algorithm reach the result in a short time. Moreover, it adapts easily to different conditions. Thus, let $B(n)$ be the time complexity of ADSMTDE in the worst case where n is data points. Then, we have $B(n) = n^2 + n^2 + n = 2n^2 + n = O(n^2)$. On the other hand, as the complexity of GMM is $O(pdn^3)$[68] for p Gaussian components and d dimensions, the auto-encoder is $O(n^2)$ and the complexity time of N2D is $O(pdn^3) + O(n^2)$ which shows that the complexity of ADSMTDE is less than N2D.

5.3. Results

5.3.1. Comparison of clustering algorithms

A comparison of the proposed method with a set of clustering algorithms on six datasets based on AC and NMI metrics is shown in Table 1.

Table 1
Comparison of our method in both automatic (has no prior information of cluster number) and non-automatic (has prior information of cluster number) modes with conventional clustering algorithms and the latest deep clustering algorithms based on the best values of AC and NMI.

Types	Methods	MNIST		USPS		Fashion		Pendigits		HAR		CIFAR10		Tiny ImageNet	
		AC	NMI	AC	NMI	AC	NMI	AC	NMI	AC	NMI	AC	NMI	AC	NMI
Conventional (Non-automatic)	k-means [59]	53.2	45	66.8	62.6	47.4	51.2	66.6	68.1	59.9	58.8	22.9	8.7	2.5	6.5
	SC [60]	68	75.9	65.6	79.6	55.1	63	72.4	78.4	53.8	74.1	24.7	10.3	2.2	6.3
	GMM [61]	38.9	33.3	56.2	54	46.3	51.4	67.3	68.2	58.5	64.8	–	–		
	DE [69]	41.41	28.53	56.4	46.9	47.83	42.74	68.33	63.5	64.53	64.07	–	–		
	MTDE [37]	46.93	34.08	57.5	43.9	51.35	46.29	68.6	62.7	67.61	64.24	32.61	28.24	2.44	3.76
Deep (Non-automatic)	DeepCluster [70]	79.7	66.1	56.2	54	54.2	51	–	–	–	–	–	–	–	–
	DCN [21]	83	81.0	68.8	68.3	50.1	55.8	72	69	–	–	–	–	–	–
	DEC [18]	86.3	83.4	76.2	76.7	51.8	54.6	70.1	67.8	56.5	58.4	–	–	–	–
	IDEC [39]	88.1	86.7	76.1	78.5	52.9	55.7	78.4	72.3	64.2	60.9	–	–	–	–
	SR-k-means [71]	93.9	86.6	90.1	91.2	50.7	54.8	–	–	–	–	–	–	–	–
	VaDE [23]	94.5	87.6	56.6	51.2	57.8	63	–	–	–	–	–	–	–	–
	ClusterGAN [72]	96.4	92.1	–	–	63	64	77	73	–	–	–	–	–	–
	JULE [24]	96.4	91.3	95	**91.3**	56.3	60.8	–	–	–	–	27.2	19.2	3.3	10.2
	DEPICT [22]	96.5	91.7	89.9	90.6	39.2	39.2	–	–	–	–	–	–	–	–
	DBC [20]	96.4	91.7	–	–	–	–	–	–	–	–	–	–	–	–
	DAC [73]	97.8	93.5	–	–	–	–	–	–	–	–	52.2	39.6	6.6	19.0
	ASPC-DA [74]	**98.8**	**96.6**	**98.2**	**95.1**	59.1	65.4	–	–	–	–	–	–	–	–
	N2D [48]	**97.9**	**94.2**	**95.8**	90.1	**67.2**	**68.4**	**88.5**	**86.3**	**80.1**	**68.3**	–	–	–	–
	EDESC [51]	–	–	–	–	–	–	–	–	–	–	46.4	62.7	–	–
	DSMTDE	**98.61**	**93.67**	**95.63**	**98.09**	**67.3**	**66.96**	**90.36**	**87.3**	**81.71**	**73.2**	49.53	46.47	4.15	3.31
Conventional (Automatic)	AMTDE	40.46	27.97	54.26	41.81	45.92	38.67	61.55	56.47	46.35	35.25	21.48	16.40	1.05	1.32
	DPSO [62]	36.65	23.25	51.43	37.93	39.13	35.49	57.18	51.75	47.55	36.54	–	–	–	–
	PSOAC [63]	36.72	24.36	52.13	39.07	41.04	34.07	57.98	54.71	47.95	36.81	–	–	–	–
	KM [64]	36.19	20.98	48.46	37.31	40.64	32.80	56.14	50.28	45.26	30.67	–	–	–	–
	DBSCAN [29]	21.06	11.47	27.25	18.97	16.17	13.13	42.48	21.02	31.05	22.78	–	–	–	–
Deep (Automatic)	ADSMTDE	88.92	90.82	80.99	78.72	**64.88**	**67.44**	**82.31**	**78.78**	**69.6**	**64.7**	40.18	37.91	3.37	2.80

The compared results are either reported in relevant papers or reproduced (k-means, SC, GMM, DE, MTDE, DSMTDE, KM, PSOAC, DPSO, AMTDE, and DBSCAN) in this work to perform a comprehensive comparison of our proposed clustering method with existing automatic and non-automatic clustering methods. Those with (-) do not have results in the relevant dataset.

For comparison, we examine the proposed algorithm in two versions. The first is Deep sparse Multi-Trial vector-based Differential Evolution (DSMTDE), which is aware of the number of clusters and is not automatic, and the second is Automatic Deep Sparse clustering technique based on an evolutionary algorithm called Multi-Trial vector-based Differential Evolution (ADSMTDE), which is completely unsupervised and automatic.

The obtained results in Table 1 indicate that the DSMTDE is one of the top three algorithms. In all compared deep clustering algorithms, although they have a strong structure for finding hidden data representation, simple clustering methods such as k-means have been used for clustering. Algorithms ASPCDA and N2D are among the top three algorithms along with the proposed algorithm.

ASPC-DA uses two deep networks. First, an autoencoder is trained and a second autoencoder is jointly trained with a clustering and reconstruction loss. However, we use UMAP instead of second deep network that has less complexity. Moreover, N2D uses a shallow clustering leading to an increase in the chances of premature convergence due to the lack of balance between exploitation and exploration. However, the proposed method combines various search strategies in clustering to cope with different problems in search space and offers a better solution than N2D because of the diversity among solutions. This has made the DSMTDE superior.

In the DSMTDE, despite a simple AE for obtaining features, it obtained excellent results by using different search strategies in MTDE clustering, which have potential capabilities in global and local search. Also, due to the benefit of a population distribution mechanism, it has been able to balance the search strategies to prevent early convergence.

Moreover, when the proposed algorithm performs clustering completely automatically using the incremental technique (ADSMTDE), it outperformed all automatic clustering methods. ADSMTDE improved the performance of MTDE clustering by 41.99% in MNIST, 23.49% in USPS, and 13.53% in Fashion 13.71% in Pendigits, 1.99% in HAR, 7.57% in CIFAR-10 and 0.93% in Tiny ImageNet.

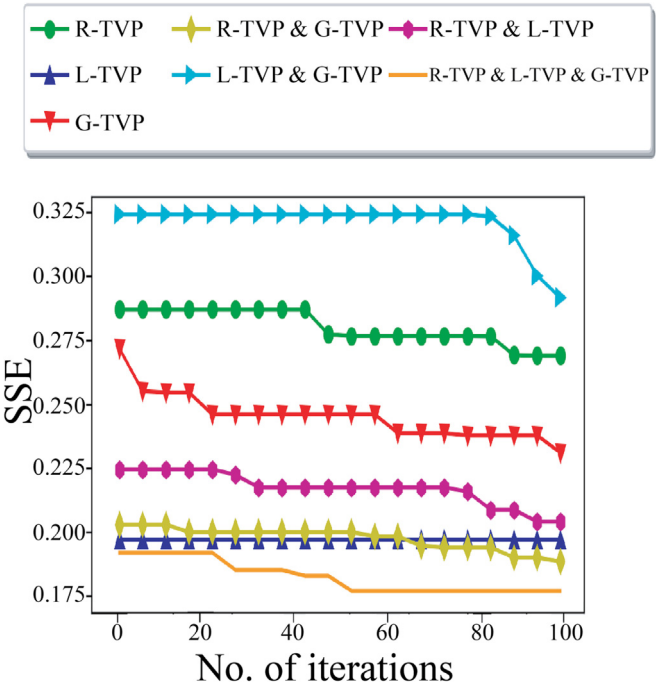


Fig. 2. The impact of distribution policy and strategies on convergence on USPS dataset.

Compared to all conventional and deep clustering algorithms that have prior knowledge of the number of clusters, in three datasets out of seven datasets including Fashion, Pendigits, and HAR, ADSMTDE was among the top three cluster approaches. It is worth mentioning that the results obtained for CIFAR-10 and Tiny ImageNet datasets compared to other datasets, the performance of ADSMTDE was weak. Although Multi-Layer Perception (MLP) autoencoder is used to extract useful features from input data, it may not be able to find suitable features from a color image because it is a relatively simple model and may not have the capacity or complexity to effectively process and analyze the multiple channels of color data. There are color images with various categories in CIFAR-10 and Tiny ImageNet datasets which require the use of a network such as a convolutional neural network to obtain better results.

But in this paper, we focus on presenting a method that can perform clustering automatically. In other words, we focused on the clustering part in order to develop a general clustering method for various types of data with several strategies that can improve its performance by considering various aspects of the search, including exploration and exploitation. We also tried to solve the problem of dependence of unsupervised methods on knowing the number of clusters by presenting the incremental technique. If this method is used specifically for image clustering, networks like CNNs must be used and if it is used for sequential data, Recurrent Neural Networks (RNN) should be used. These networks lead to the extraction of more suitable features from images and sequences, respectively.

5.3.2. Effect of MTDE strategies

Fig. 2 demonstrates the effect of three MTDE strategies based on convergence rates on USPS dataset. The combination of all three strategies resulted in a good convergence with the lowest SSE value due to simultaneous global and local searches and maintaining a balance between them.

Table 2 indicates the error in estimating the number of clusters using different methods calculated based on [75]. Incremental technique has outperformed all methods in all datasets. In this method, instead of clustering all data points, the clusters are selected gradually based on the degree of results improvement. But in cluster validity methods, not only they performed poorly in detecting the number of clusters, but one method was better than others in different datasets, which indicates that their results are not stable.

Table 2
Comparison of average error for detecting the number of clusters.

Dataset	USPS	Pendigits	MNIST	HAR	Fashion	CIFAR10	Tiny ImageNet
Sill [76]	2.4	2.22	2.56	2.63	3.31	3.8	9.78
CH [77]	2.96	3.06	3.07	3.31	2.98	4.69	12.37
DB [78]	2.94	3.20	3.29	3.15	4.07	4.66	21.77
Dunn [79]	4.67	4.24	4.68	5.19	4.6	6.12	12.75
GAP [80]	4.58	4.5	5.71	4.62	5.07	6.30	19.54
Elbow [81]	3.7	4.43	4.44	5.07	4.64	6.33	16.64
Ha [82]	3.33	3.18	3.81	4.57	3.37	5.19	14.98
Incremental technique (Ours)	2.18	2.02	1.154	2.64	1.414	3.46	8.61

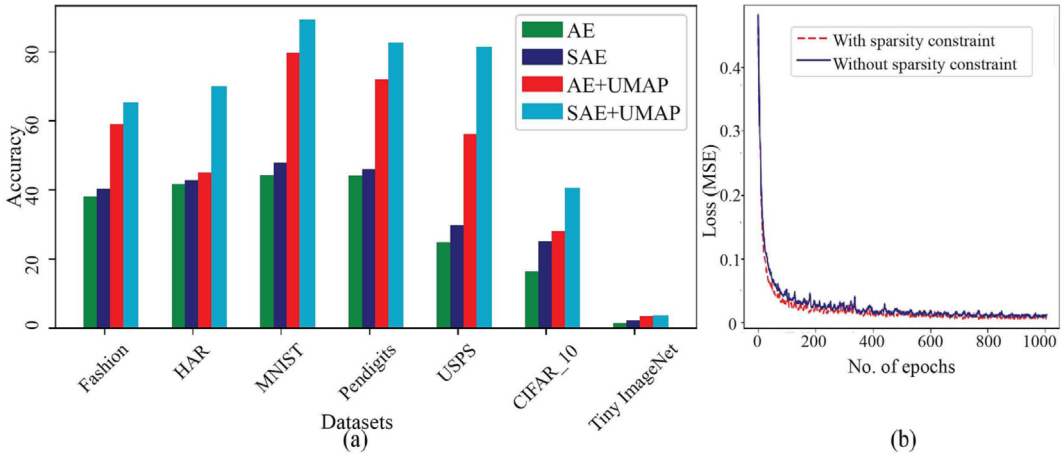
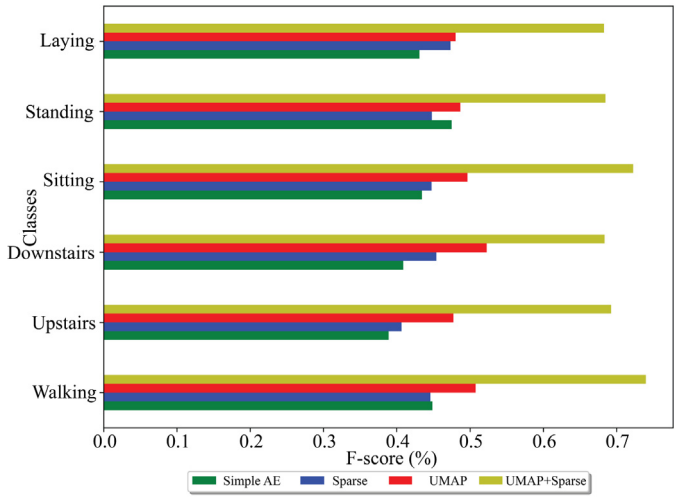


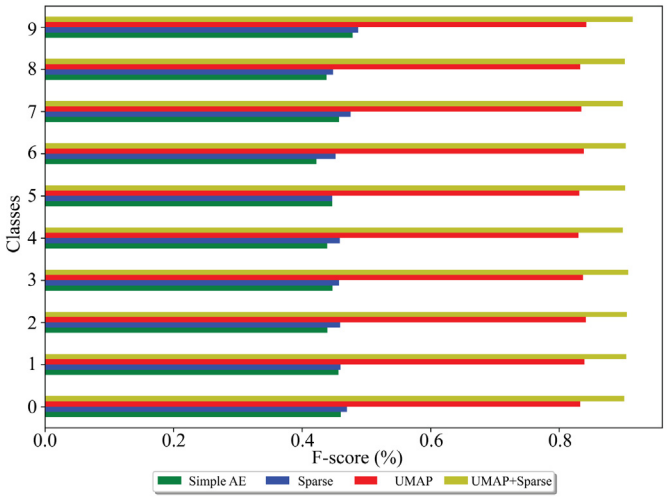
Fig. 3. Performance analysis components applied to AE based on (a) accuracy and (b) convergence rate.

Table 3
Clustering accuracy comparisons for different data representation components on various datasets. AE stands for Autoencoder without sparse and UMAP, AE + UMAP stands for Autoencoder with UMAP, AE + sparse stands for Autoencoder with sparse, and AE + UMAP + Sparse stands for Autoencoder with UMAP and sparse.

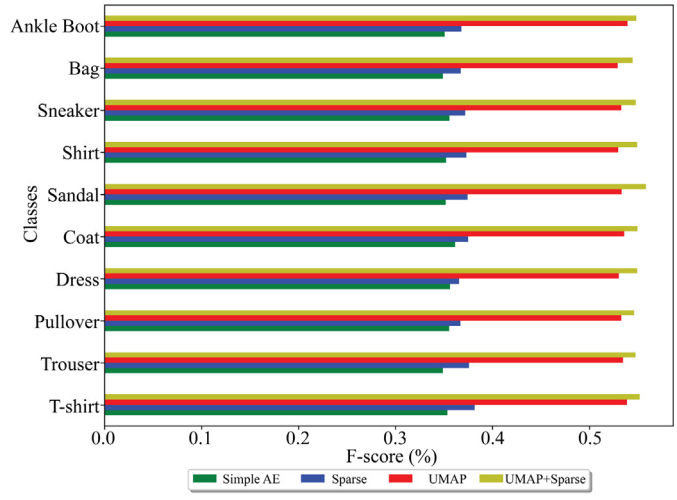
Dataset	USPS	Pendigits	MNIST	HAR	Fashion	CIFAR-10	Tiny ImageNet
AE	24.52	43.77	43.82	41.37	37.83	16.07	1.24
AE + UMAP	55.74	71.82	79.27	44.63	58.59	27.71	3.06
AE + sparse	29.46	45.65	47.52	42.42	40.02	24.80	1.91
AE + UMAP + Sparse	80.99	82.31	88.92	69.60	64.88	40.18	3.37



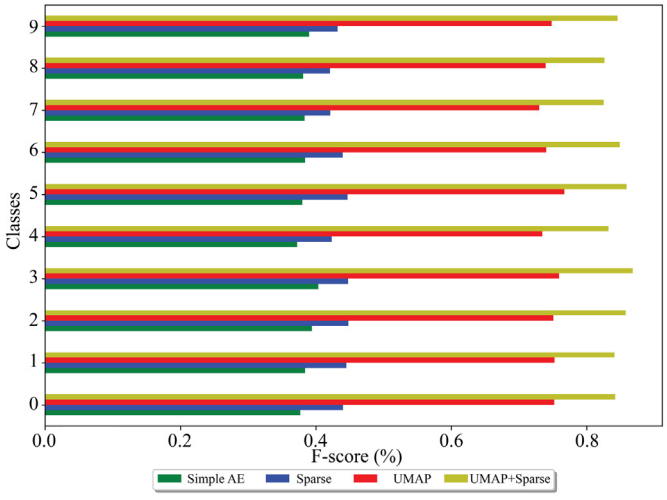
(a) F-Score for all classes in HAR



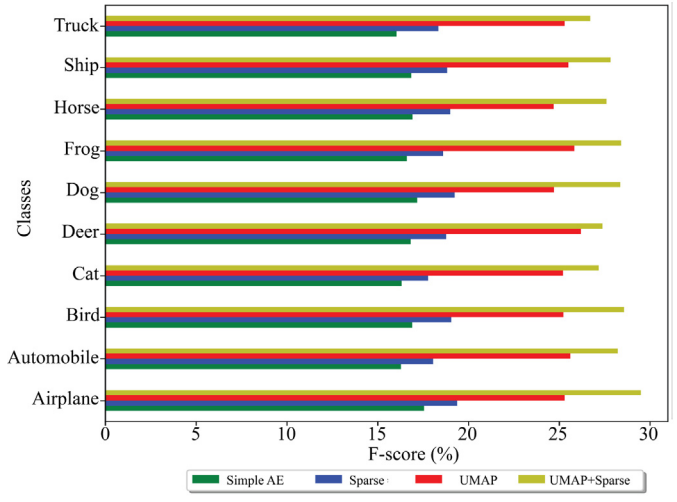
(b) F-Score for all classes in MNIST



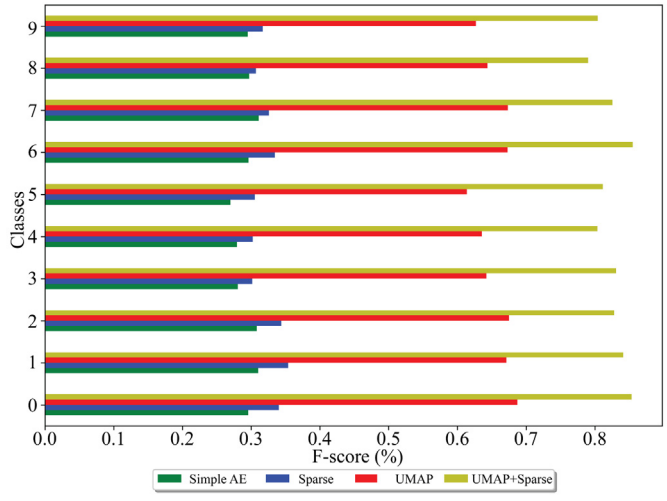
(c) F-Score for all classes in FASHION



(d) F-Score for all classes in Pendigits



(e) F-Score for all classes in CIFAR



(f) F-Score for all classes in USPS

Fig. 4. The average f-score for all classes in six datasets, obtained by different methods based on our proposed deep clustering.

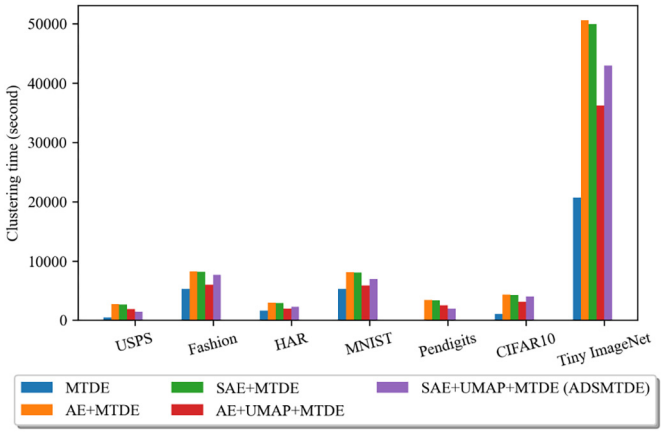


Fig. 5. The comparison of clustering time between five methods MTDE, AE + MTDE, SAE + MTDE, AE + UMAP + MTDE and SAE + UMAP + MTDE (ADSMTDE).

5.3.3. Data representation performance

Fig. 3(a) shows the effect of sparse constraint and UMAP based on obtained accuracy for each dataset. This figure shows that by using sparse constraint and UMAP in AE, the accuracy has been improved an average of 45%, 41%, and 7% in comparison with Simple AE, sparse AE, AE + UMAP respectively. Moreover, Fig. 3b shows the convergence rate of AE with and without using sparse constraint. Comparing the two methods, AE with sparse constraint has better convergence than Simple AE. This improvement is due to disabling nodes that copied input to output AE. This has led to the discovery of more important features. In addition, although AE has a great ability to find important features of data, it is weak in recognizing local data representation such as the distance between data points [48]. Since UMAP is very efficient in finding these types of features [48], applying it to the latent space has significantly increased clustering performance.

Table 3 presents a comprehensive comparison of the clustering performance achieved by different data representation components on various datasets. The table shows that each component alone has a positive

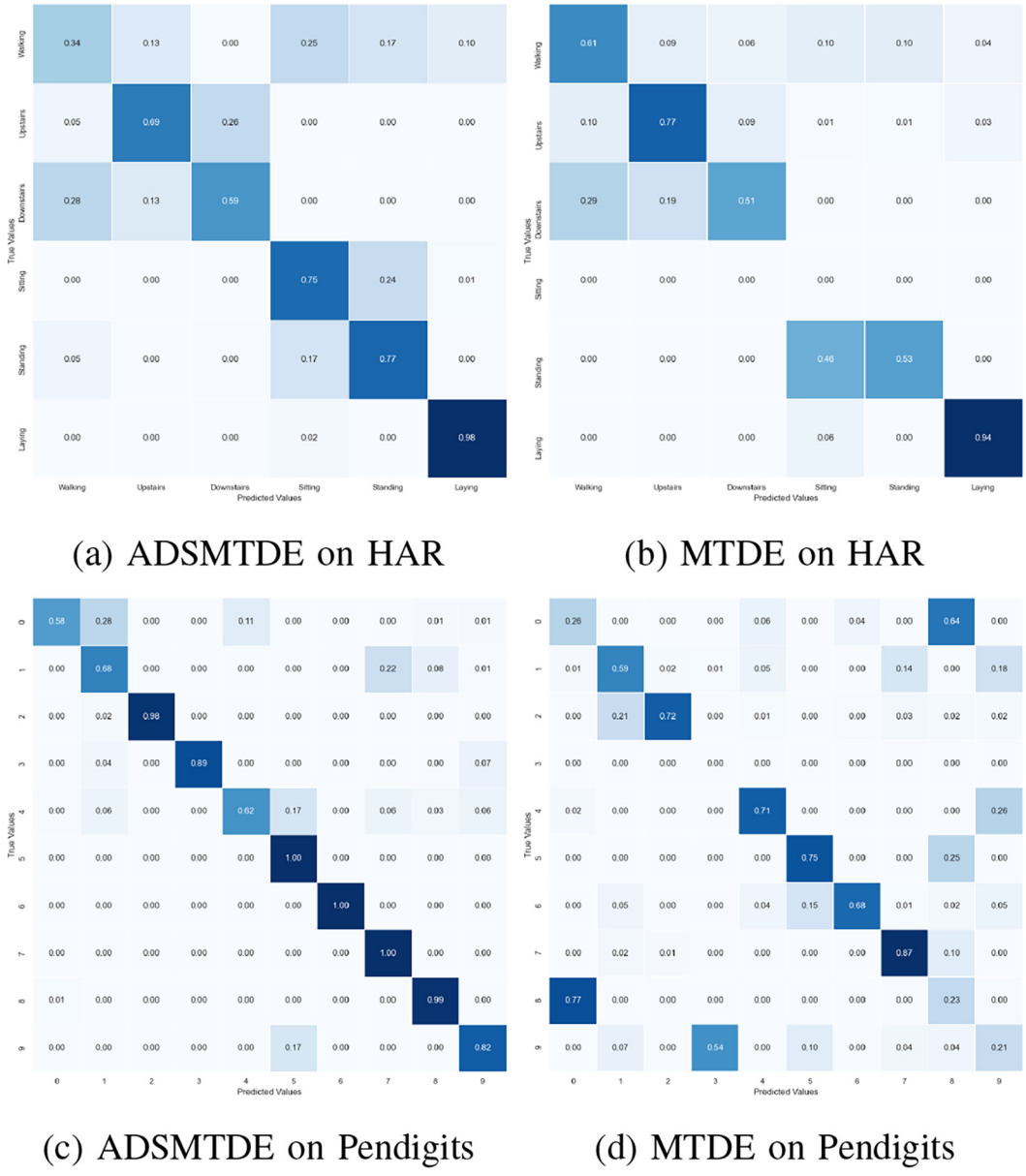


Fig. 6. Confusion matrices of the two methods ADSMTDE and MTDE on two datasets, HAR and Pendigits.

Table 4
The pretest results of tuning the parameter *final* when the *initial* value was 0.001.

Datasets	Methods	*initial* = 0.001				
		final = 0.1	*final* = 1	*final* = 2	*final* = 5	*final* = 10
MNIST	Fitness	0.9158	0.7720	0.7857	0.7857	0.8858
	Sensitivity	99.94	99.94	99.94	99.94	99.94
	Specificity	99.94	99.98	99.98	99.98	99.92
USPS	Fitness	0.8297	0.7562	0.7499	0.8397	0.8499
	Sensitivity	70.63	70.49	70.63	70.49	70.49
	Specificity	100	100	100	100	100
Fashion	Fitness	0.6158	0.6743	0.6158	0.7158	0.6843
	Sensitivity	75.911	75.911	70.773	76.076	60.052
	Specificity	84.495	83.912	78.519	84.348	
Pendigits	Fitness	0.7884	0.7584	0.6584	0.7756	0.8135
	Sensitivity	97.542	97.586	97.619	97.642	97.617
	Specificity	97.437	97.751	97.403	97.576	97.246
HAR	Fitness	0.6652	0.6300	0.6300	0.6952	0.7052
	Sensitivity	97.481	97.556	97.433	97.501	97.360
	Specificity	92.394	92.136	92.298	92.270	91.712
CIFAR10	Fitness	0.9607	0.8955	0.9229	0.9327	0.9327
	Sensitivity	84.49	83.91	78.51	84.34	66.78
	Specificity	60.23	60.14	56.05	60.98	47.24
Tiny ImageNet	Fitness	0.7177	0.7352	0.7580	0.6665	0.7123
	Sensitivity	68.34	73.64	82.61	71.68	83.89
	Specificity	68.28	68.09	69.91	69.38	66.28

impact on the clustering performance, but combining all three components yields the best results. Using UMAP embedding in combination with AE helps in finding a better clusterable representation of data, while incorporating sparse constraints prevents the extraction of redundant features, leading to better clustering accuracy. The improvement achieved by the combination of these components is particularly significant on most datasets. However, the results show that the performance of the approach varies across different datasets. For instance, the MNIST dataset achieves a higher clustering accuracy than the other datasets when using all three components. This result may be due to the simplicity of the dataset's images, which are grayscale and have a low resolution. On the other hand, the Tiny ImageNet dataset, which comprises color images, achieves the lowest clustering accuracy across all four configurations. This result may be due to the complexity of the dataset's images, which have a high dimensionality and a more diverse range of classes and images, making it more challenging to find a good

Table 5
The pretest results of tuning the parameter *initial* when the *final* value was 2.

Datasets	Methods	*final* =2				
		initial = 0.001	*initial* = 0.01	*initial* = 0.1	*initial* = 1	*initial* = 2
MNIST	Fitness	0.6565	0.7419	0.8900	0.8454	0.7454
	Sensitivity	97.56	97.69	97.44	97.46	97.56
	Specificity	97.71	97.37	97.08	97.16	97.77
USPS	Fitness	0.6986	0.8386	0.6986	0.8286	0.7986
	Sensitivity	97.34	97.57	97.44	97.56	97.577
	Specificity	91.86	92.71	92.20	92.11	92.445
Fashion	Fitness	0.6208	0.7403	0.6658	0.6958	0.7142
	Sensitivity	84.153	84.204	83.809	84.234	83.969
	Specificity	60.652	60.892	59.776	59.132	60.860
Pendigits	Fitness	0.7454	0.6454	0.6454	0.7857	0.7754
	Sensitivity	97.34	97.57	97.44	97.56	97.57
	Specificity	91.86	92.71	92.20	92.11	92.44
HAR	Fitness	0.6394	0.7059	0.6059	0.6059	0.6794
	Sensitivity	97.56	97.69	97.44	97.46	97.56
	Specificity	97.71	97.37	97.08	97.16	97.77
CIFAR10	Fitness	0.9051	0.9665	0.9838	0.8914	0.9371
	Sensitivity	62.48	60.80	60.41	60.35	58.93
	Specificity	94.37	94.65	94.95	94.79	94.50
Tiny ImageNet	Fitness	0.6742	0.7582	0.7139	0.7309	0.8141
	Sensitivity	75.94	72.40	72.19	72.22	72.72
	Specificity	79.03	72.04	78.41	75.86	79.77

clustering solution. Therefore, more complex network architectures may be required to improve the clustering performance on color image datasets.

Fig. 4 shows the effect of each component in the proposed method, which includes sparse constraint and UMAP, based on obtained F-score for each class of the different datasets. This figure shows that by using sparse constraint and UMAP in AE (Auto-encoder), the f-score of each class has been improved an average of 45%, 41%, and 7% in comparison with Simple AE, sparse AE, AE + UMAP. This improvement is due to disabling nodes that copied input to output AE. This has led to the discovery of more important features. In addition, although AE has a great ability to find important features of data, it has a weakness in recognizing local data representation such as the distance between data points [48]. Since UMAP is very efficient in recognizing local data representation which AE is poor at these types of features [48], applying it to the latent space has significantly increased clustering performance.

Fig. 5 shows the comparison of clustering time between five methods MTDE, AE + MTDE, SAE + MTDE, AE + UMAP + MTDE and SAE + UMAP + MTDE (ADSMTDE). The clustering time varies widely across the different methods and datasets. The MTDE generally has the shortest clustering time. This is likely because MTDE is a relatively simple clustering method compared to the others, and does not involve any additional preprocessing steps like auto-encoding or UMAP. The AE + MTDE and SAE + MTDE methods have longer clustering times, respectively. This is likely due to the additional step of training an auto-encoder or sparse auto-encoder before applying the MTDE clustering algorithm. These methods can be more computationally intensive

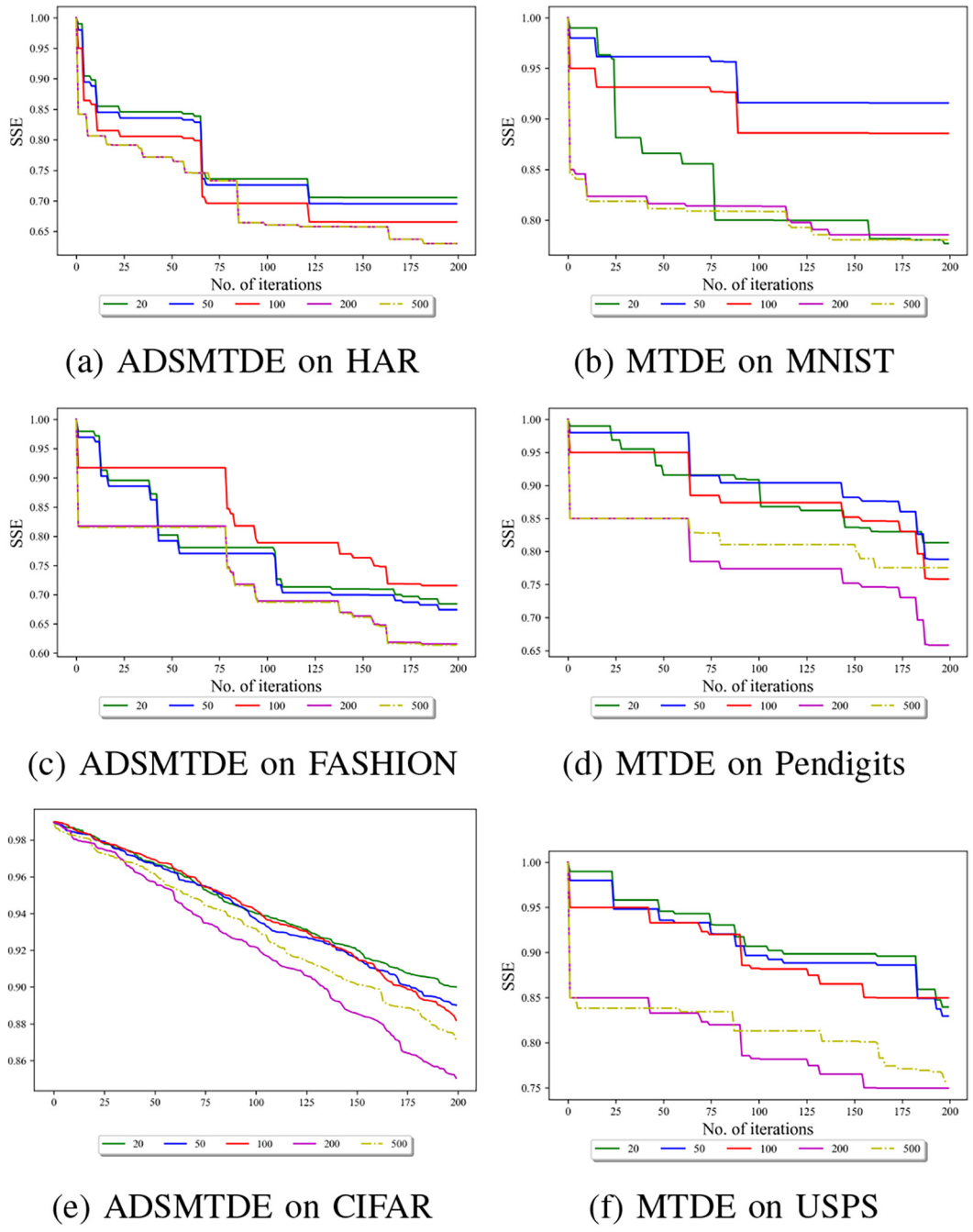


Fig. 7. llustration of comparing convergence with different population sizes of ADSMTDE vs. MTDE on datasets, HAR, MNIST, Fashion, Pendigits, CIFAR10 and USPS.

due to the added complexity of training neural networks. AE + UMAP + MTDE has a lower clustering time than AE + MTDE and SAE + MTDE. The reason for this is attributed to the added step of applying UMAP to reduce the dimensionality of the data before applying the MTDE clustering algorithm. SAE + UMAP + MTDE (ADSMTDE) includes both the sparse auto-encoder and UMAP steps, which can make it more computationally intensive than MTDE alone. However, it still has a relatively short clustering time compared to the other methods, likely due to the time efficiency gained from the use of UMAP.

Overall, the clustering time results suggest that MTDE is the most computationally efficient clustering method among those considered, while the other methods have longer clustering times due to additional preprocessing steps. However, it is important to note that ADSMTDE has been shown to achieve better results, as indicated in Table 1. Our study shows that MTDE, despite spending less time on clustering compared to ADSMTDE, performs poorly in clustering due to its inability to extract meaningful information from the data. As indicated in Table 1, MTDE has the worst clustering results among all the algorithms considered in Fig. 5. Hence, we maintain that our proposed method, ADSMTDE, is more time-efficient than other methods while still achieving competitive clustering performance.

5.3.4. Comparison of ADSMTDE and MTDE

Fig. 6 shows a comparison of the confusion matrix between ADSMTDE and MTDE in two datasets HAR (Fig. 6 and Pendigits (Fig. 6c and d). From the figure, it can be seen that by using AE for learning the data representation in MTDE, the confusion between the clusters was drastically reduced. Moreover, ADSMTDE compared to MTDE has obtained these results without prior knowledge of the number of clusters which shows that the incremental technique has performed well in identifying clusters.

5.3.5. Parameter tuning in proposed algorithm

One of the important issues in obtaining the best results is adjusting the parameters used in the proposed algorithm. Thus, we conducted pretests to fine-tune the variables of *final* and *initial* in Eq. (11) where results are shown in Tables 4 and 5, respectively. The findings are made up of the mean values of various qualitative measures, such as fitness, sensitivity and specificity.

Fitness is a measure of the quality of a solution, which we defined as the objective function value (SSE in Eq. (12)) that represents the accuracy of the proposed algorithm in solving the optimization problem. Sensitivity and specificity measure the proportion of true positive and true negative results, respectively. To fine-tune the variables of our proposed algorithm, we evaluated its performance on seven datasets for each combination of final and initial values and recorded the fitness score, sensitivity, and specificity. We repeated this process for several iterations to obtain reliable estimates of the mean values of these metrics. We then analyzed these results to identify the optimal combination of final and initial values that produced the highest fitness score and the highest values of sensitivity and specificity. Using these results, we fine-tuned the variables of the proposed algorithm, which led to better performance and improved convergence rates. The results of the pretests demonstrate that the values 2 and 0.001 used for *final* and *initial* can be sufficiently adjusted.

Additionally, Fig. 7 depicts the convergence of the proposed algorithm on different datasets with various population size. From the obtained results, it can be seen that the proposed algorithm converges better when the population size is 200.

One way to assess the convergence of the proposed algorithm is by analyzing the trend of the fitness curve over the iterations. In Fig. 7, the fitness curve for each population size is plotted against the number of iterations. By observing the graph, it becomes apparent that the fitness curve for a population size of 200 exhibits faster convergence and achieves a lower value compared to the other population sizes. This observation suggests that the proposed algorithm performs better with a population size of 200. Additionally, this finding is consistent with prior research [83] that has demonstrated how a smaller population size can result in improved convergence rates and more diverse solutions.

6. Conclusions

A new automatic deep clustering method based on an evolutionary algorithm has been presented. The proposed method applied sparsity and manifold learning on AE to enhance feature learning and retain the local structure of data. An evolutionary algorithm that excelled in exploring and exploiting was used for clustering. To solve the problem of determining the number of clusters, an incremental method is employed to cluster data without knowing the number of clusters. Experiments on a variety of datasets, including image and time-series datasets, were carried out to show the performance of the proposed method in comparison with recent SOTA methods. The results show that applying both sparse constraint and UMAP to AE increased the clustering efficiency and improved the clustering performance by 45% compared to simple AE. It was validated that the proposed approach was among the top 3 current SOTA clustering methods in the majority of datasets when the number of clusters was known. Also, promising results were obtained when clustering was done automatically and outperformed many SOTA clustering methods.

As future work, complex networks like Convolutional Neural Networks (CNNs) can be used to better represent color images. One of the essential steps to getting a good representation of the input patterns for clustering is feature extraction. The proposed method is a general clustering method that has adopted an artificial neural network with clustering. If it is to be used for image or video clustering, CNNs should be used to better extract the features of the images and videos. Moreover, if it is to be used for sequential data such as genomic data, RNNs should be used. CNNs and RNNs can extract informative characteristics from images and sequence data respectively. These networks minimize the high dimensionality of data without sacrificing any information that is useful for clustering. Moreover, the effect of multi-objectives on the performance of clustering can be investigated.

CRediT authorship contribution statement

Parham Hadikhani: Project-administration, Conceptualization, Methodology, Software, Validation, Formal-analysis, Investigation, Resources, Writing-original-draft, Writing-review-editing, Visualization. **Daphne Teck Ching Lai:** Supervision, Writing-review-editing. **Wee-Hong Ong:** Supervision, Writing-review-editing. **Mohammad H. Nadimi-Shahraki:** Writing-review-editing.

Data availability

No data was used for the research described in the article.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Parham Hadikhani reports financial support was provided by University of Brunei Darussalam.

Acknowledgment

This work was supported by Grant UBD/RSCH/1.11/ FICBF(b)/2019/001 from Universiti Brunei Darussalam.

References

[1] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, J. Long, A survey of clustering with deep learning: From the perspective of network architecture, IEEE Access 6 (2018) 39 501–39 514.
[2] V. Kumar, W. Reinartz, Customer Relationship Management: Concept, Strategy, and Tools, Springer, 2016.
[3] A. Jain, R. Dubes, Algorithms for Clustering Data, Prentice-Hall, 1988.
[4] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 603–619.
[5] P. Felzenszwalb, D. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vis. 59 (2) (2004) 167–181.
[6] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. (CSUR) 41 (3) (2009) 15.
[7] M. Breunig, H. Kriegel, R. Ng, J. Sander, Lof: Identifying density-based local outliers, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, ACM 2000, pp. 93–104.
[8] P. Hadikhani, D.T.C. Lai, W.-H. Ong, Flexible multi-objective particle swarm optimization clustering with game theory to address human activity recognition fully unsupervised, 2022.
[9] F. Ricci, L. Rokach, B. Shapira, Recommender Systems Handbook, Springer, 2015.
[10] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM 2008, pp. 426–434.
[11] P. Hadikhani, M. Eslaminejad, M. Yari, E. Ashoor Mahani, An energy-aware and load balanced distributed geographic routing algorithm for wireless sensor networks with dynamic hole, Wireless Netw. 26 (1) (2020) 507–519.
[12] M. Yari, P. Hadikhani, Z. Asgharzadeh, Energy-efficient topology to enhance the wireless sensor network lifetime using connectivity control, J. Telecommun. Digit. Econ. 8 (3) (2020) 68–84.
[13] M. Yari, P. Hadikhani, M. Yaghoubi, R. Nowrozy, Z. Asgharzadeh, An energy efficient routing algorithm for wireless sensor networks using mobile sensors, arXiv preprint arXiv:2103.04119, 2021.
[14] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometr. Intell. Lab. Syst. 2 (1–3) (1987) 37–52.
[15] P. Xanthopoulos, P.M. Pardalos, T.B. Trafalis, Linear discriminant analysis, Robust data mining, Springer 2013, pp. 27–33.
[16] H. Hoffmann, Kernel pca for novelty detection, Pattern Recogn. 40 (3) (2007) 863–874.
[17] M.C. Cieslak, A.M. Castelfranco, V. Roncalli, P.H. Lenz, D.K. Hartline, t-distributed stochastic neighbor embedding (t-sne): A tool for eco-physiological transcriptomic analysis, Mar. Genomics 51 (2020), 100723.
[18] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, International conference on machine learning, PMLR 2016, pp. 478–487.
[19] A. Ng, et al., Sparse autoencoder, CS294A Lect. Notes 72 (2011) (2011) 1–19.
[20] F. Li, H. Qiao, B. Zhang, Discriminatively boosted image clustering with fully convolutional auto-encoders, Pattern Recogn. 83 (2018) 161–173.
[21] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, International conference on machine learning, PMLR 2017, pp. 3861–3870.
[22] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, H. Huang, Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5736–5745.
[23] Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: An unsupervised and generative approach to clustering, arXiv preprint arXiv:1611.05148, 2016.
[24] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 5147–5156.
[25] C.-C. Hsu, C.-W. Lin, Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data, IEEE Trans. Multimed. 20 (2) (2017) 421–429.
[26] Z. Wang, S. Chang, J. Zhou, M. Wang, T.S. Huang, Learning a task-specific deep architecture for clustering, Proceedings of the 2016 SIAM International Conference on Data Mining, SIAM 2016, pp. 369–377.
[27] L. Kaufman, P.J. Rousseeuw, Finding groups in data: an introduction to cluster analysis, vol. 344, , John Wiley & Sons, 2009.
[28] J. McLachlan Geoffrey, The em algorithm and extensions/geoffrey j. mclachlan, thriyambakam krishnan, 1997.
[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: kdd, vol. 96, no. 34, 1996, pp. 226–231.
[30] C.-F. Tsai, H.-C. Wu, C.-W. Tsai, A new data clustering approach for data mining in large databases, Proceedings International Symposium on Parallel Architectures, Algorithms and Networks. I-SPAN'02, IEEE 2002, pp. 315–320.
[31] W. Peizhuang, Pattern recognition with fuzzy objective function algorithms (james c. bezdek), SIAM Rev. 25 (3) (1983) 442.
[32] P. Hadikhani, P. Hadikhani, An adaptive hybrid algorithm for social networks to choose groups with independent members, Evol. Intel. 13 (4) (2020) 695–703.
[33] P. Hadikhani, D.T.C. Lai, W.-H. Ong, A novel skeleton-based human activity discovery technique using particle swarm optimization with gaussian mutation, arXiv preprint arXiv:2201.05314, 2022.
[34] T. Lei, X. Jia, Y. Zhang, L. He, H. Meng, A.K. Nandi, Significantly fast and robust fuzzy c-means clustering algorithm based on morphological reconstruction and membership filtering, IEEE Trans. Fuzzy Syst. 26 (5) (2018) 3027–3041.
[35] B.A. Hassan, T.A. Rashid, S. Mirjalili, Formal context reduction in deriving concept hierarchies from corpora using adaptive evolutionary clustering algorithm star, Complex Intell. Syst. (2021) 1–16.
[36] Y. Li, H. Shi, L. Jiao, R. Liu, Quantum evolutionary clustering algorithm based on watershed applied to sar image segmentation, Neurocomputing 87 (2012) 90–98.
[37] P. Hadikhani, D.T.C. Lai, W.-H. Ong, M.H. Nadimi-Shahraki, Improved data clustering using multi-trial vector-based differential evolution with gaussian crossover, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2022, pp. 487–490.
[38] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341.
[39] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: Ijcai, 2017, pp. 1753–1759.
[40] W. Menapace, S. Lathuilière, E. Ricci, Learning to cluster under domain shift, Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16, Springer 2020, pp. 736–752.
[41] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, L. Van Gool, Scan: Learning to classify images without labels, European Conference on Computer Vision, Springer 2020, pp. 268–285.
[42] N. Astorga, P. Huijse, P. Protopapas, P. Estévez, Mpcc: Matching priors and conditionals for clustering, European Conference on Computer Vision, Springer 2020, pp. 658–677.
[43] J. Zhao, D. Lu, K. Ma, Y. Zhang, Y. Zheng, Deep image clustering with category-style representation, European Conference on Computer Vision, Springer 2020, pp. 54–70.
[44] C. Niu, J. Zhang, G. Wang, J. Liang, Gatcluster: Self-supervised gaussian-attention network for image clustering, European Conference on Computer Vision, Springer 2020, pp. 735–751.
[45] J. Huang, S. Gong, X. 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.
[46] Y. Li, P. Hu, Z. Liu, D. Peng, J.T. Zhou, X. Peng, Contrastive clustering, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 10, 2021, pp. 8547–8555.
[47] M. Sadeghi, H. Hojjati, N. Armanfard, C3: Cross-instance guided contrastive clustering, arXiv preprint arXiv:2211.07136, 2022.
[48] R. McConville, R. Santos-Rodriguez, R.J. Piechocki, I. Craddock, N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding, 2020 25th International Conference on Pattern Recognition (ICPR), IEEE 2021, pp. 5145–5152.
[49] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks, Adv. Neural Inf. Process. Syst. 30 (2017).
[50] T. Zhang, P. Ji, M. Harandi, R. Hartley, I. Reid, Scalable deep k-subspace clustering, Asian Conference on Computer Vision, Springer 2018, pp. 466–481.
[51] J. Cai, J. Fan, W. Guo, S. Wang, Y. Zhang, Z. Zhang, Efficient deep embedded subspace clustering, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 1–10.
[52] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, Y. Kluger, Spectralnet: Spectral clustering using deep neural networks, arXiv preprint arXiv:1801.01587, 2018.
[53] T. Higuchi, K. Kinoshita, M. Delcroix, K. Zmolíková, T. Nakatani, Deep clustering-based beamforming for separation with unknown number of sources, Interspeech (2017) 1183–1187.
[54] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, Acm sigkdd Explor. Newslett. 6 (1) (2004) 90–105.
[55] R. Shi, J. Ji, C. Zhang, Q. Miao, Boosting sparsity-induced autoencoder: A novel sparse feature ensemble learning for image classification, Int. J. Adv. Rob. Syst. 16 (3) (2019), 1729881419853471.
[56] L. McInnes, J. Healy, J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, arXiv preprint arXiv:1802.03426, 2018.
[57] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, 2013 IEEE congress on evolutionary computation, IEEE 2013, pp. 71–78.
[58] M.R. Murty, J. Murthy, P. Reddy, A. Naik, S. Satapathy, et al., Homogeneity separateness: a new validity measure for clustering problems, ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I, Springer 2014, pp. 1–10.
[59] S. Lloyd, Least squares quantization in pcm, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[60] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, in: Advances in neural information processing systems, 2002, pp. 849–856.
[61] D.A. Reynolds, Gaussian mixture models, Ency. Biom. 741 (659–663) (2009).
[62] M.G. Omran, A. Salman, A.P. Engelbrecht, Dynamic clustering using particle swarm optimization with application in image segmentation, Pattern Anal. Appl. 8 (4) (2006) 332–344.
[63] Y. Kao, C.-C. Chen, Automatic clustering for generalised cell formation using a hybrid particle swarm optimisation, Int. J. Prod. Res. 52 (12) (2014) 3466–3484.
[64] J. MacQueen, Classification and analysis of multivariate observations, in: 5th Berkeley Symp. Math. Statist. Probability, 1967, pp. 281–297.
[65] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[66] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[67] Y. Le, X. Yang, Tiny imagenet visual recognition challenge, CS 231N 7 (7) (2015) 3.
[68] R.C. Pinto, P.M. Engel, A fast incremental gaussian mixture model, PloS One 10 (10) (2015), e0139931.
[69] K.V. Price, Differential evolution, Handbook of optimization, Springer 2013, pp. 187–214.

[70] M. Caron, P. Bojanowski, A. Joulin, M. Douze, Deep clustering for unsupervised learning of visual features, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 132–149.
[71] M. Jabi, M. Pedersoli, A. Mitiche, I.B. Ayed, Deep clustering: On the link between discriminative models and k-means, IEEE Trans. Pattern Anal. Mach. Intell. 43 (6) (2019) 1887–1896.
[72] S. Mukherjee, H. Asnani, E. Lin, S. Kannan, Clustergan: Latent space clustering in generative adversarial networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, 2019, pp. 4610–4617.
[73] J. Chang, L. Wang, G. Meng, S. Xiang, C. Pan, Deep adaptive image clustering, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5879–5887.
[74] X. Guo, X. Liu, E. Zhu, X. Zhu, M. Li, X. Xu, J. Yin, Adaptive self-paced deep clustering with data augmentation, IEEE Trans. Knowl. Data Eng. 32 (9) (2019) 1680–1693.
[75] P. Hadikhani, D.T.C. Lai, W.-H. Ong, Human activity discovery with automatic multi-objective particle swarm optimization clustering with gaussian mutation and game theory, IEEE Trans. Multimed. (2023).
[76] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[77] T. Caliński, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat.-Theory Methods 3 (1) (1974) 1–27.
[78] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 2 (1979) 224–227.
[79] J.C. Dunn, A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters, 1973.
[80] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, J. R. Stat. Soc.: Ser. B (Stat. Methodol.) 63 (2) (2001) 411–423.
[81] R.L. Thorndike, Who belongs in the family, Psychometrika, Citeseer, 1953.
[82] J.A. Hartigan, Clustering algorithms, John Wiley & Sons Inc, 1975.
[83] A.E. Eiben, J. Smith, From evolutionary computation to the evolution of things, Nature 521 (7553) (2015) 476–482.